Notes for

Image Processing and Vision

Instituto Superior Técnico

Jorge S. Marques

IST / ISR

2007/2008

Contents

I Basics **9**

1 Images **10**

1.1 What is an image? 10

1.2 Discrete Images 12

1.3 Continuous Images 13

1.4 Color images 16

2 Filtering **20**

2.1 Introduction 20

2.2 Linear Filtering 21

2.3 Frequency Analysis 25

2.4 Matrix Notation 28

2.5 Median Filter 30

2.6 Order Filters 33

3 Image Alignment **36**

3.1 Introduction 36

3.2 Geometric Transformations 38

3.3 Aligning sets of points 39

3.4 Alignment without marks* 47

3.5 Discussion 50

4 Image Representation and Models **53**

4.1 Multi-scale representations 53

4.2 Scale-Space and Gaussian Pyramid 53

4.3 Haar Representation 55
4.4 Wavelet Transform 59
4.4.1 Multiresolution Approximation 59
4.4.2 Fast Wavelet Transform 61
4.4.3 Representation of discrete images 62
4.5 Probabilistic Models 63
4.6 First and second order histograms 64
4.7 Wavelet models 66
4.8 Markov Random Fields 67
4.8.1 Gibbs Random Fields 68
4.8.2 Equivalence of MRF and GRF 70
4.8.3 Simulation 70
4.8.4 Optimization 71
4.8.5 Parameter Estimation 73
4.9 Appendix - Fast Wavelet Transform 74

5 Learning **77**
5.1 Introduction 77
5.2 Motion Prediction 77
5.3 Texture classification 80
5.4 Parameter estimation 83
5.5 Unsupervised Texture Classification 86
5.6 Dealing with Arbitrary Shapes 88

II Image Processing **91**

6 Image Enhancement and Restoration **92**
6.1 Introduction 92
6.2 Inverse problems 93
6.3 Degradation Model 93
6.4 The naive approach 94
6.5 Regularization Methods 95
6.6 Edge preserving methods 97
6.7 Bayesian Methods 100

6.7.1 Denoising with wavelets . . . 102
6.8 Moisaicing and Superresolution . . . 102

7 Image Segmentation **103**
7.1 Introduction . . . 103
7.2 Problem Formulation . . . 104
7.3 Thresholding . . . 105
7.4 Pixel Classification . . . 106
7.5 Joint Pixel Classification . . . 109
7.6 Watershed Transform . . . 110
7.7 Active Contours . . . 113

8 Object Recognition **120**
8.1 Introduction . . . 120
8.2 Silluette Analysis . . . 122
8.3 Ajuste de template . . . 126
8.4 Detecção Probabilística . . . 126
8.5 Reconhecimento com PCA . . . 128
8.6 Modelos de constelação . . . 133

III Vision **136**

9 Camera Model **137**
9.1 Introduction . . . 137
9.2 Perspective Projection . . . 138
9.3 Extrinsic Parameters . . . 140
9.4 Intrinsic Parameters . . . 142
9.5 Camera Model in Cartesian Coordinates . . . 143
9.6 Points, lines and planes . . . 144
9.7 Camera Calibration . . . 146

10 Shape and Motion Estimation **150**
10.1 Introduction . . . 150
10.2 Epipolar Geometry . . . 151
10.3 Essential Matrix . . . 153

10.4 Estimation of the Essential Matrix 155
10.5 Shape and Motion estimation (Calibrated camera) 156

Foreword

Most of the information available in modern societies is visual. We have huge amounts of images and videos obtained by a wide variety of sensors: scanners, video cameras, medical imaging systems (e.g., CT, MRI, ultrasound, SPECT, PET), radar, sonar or satellite systems.

Each type of images has its own properties and raises new challenges since it measures a small part of the world in its own way. For example different image modalities (MRI, SPECT) reveal complementary aspects of the patient (anatomy and function) useful for diagnosis. Satellite images convey information about the land cover. Despite this diversity there are common goals. The common goal is to understand the information conveyed by each type of image and to devise methods to extract this information and fuse information from multiple images.

The areas of Image Processing and Computer Vision deal with a wide range of problems which can be grouped in terms of two main questions:

Problem 1: How can we modify images in order to enhance their properties?

Problem 2: How can we extract information from images in an automatic way?

In the first case, the outputs are images. Given one or several input images, we wish to modify them in order to enhance information (e.g., remove noise, correct a geometric distortion, compare satellite images obtained at different years). In the second problem, the output is a description of the image content (e.g., roads, people, human organs, 3D shape of a scene).

Problem 1 is the key problem in *Image Processing* and Problem 2 is the key problem in *Image Analysis and Vision.* The difference between Image Analysis and Vision is not always clear. In Image Analysis we are interested in describing the image content (e.g., detection of faces in images) while in Computer Vision we wish to extract 3D information about the real world (e.g., estimation of 3D shape and motion) and control it (e.g., robot guidance).

A few examples are listed below.

Problems:

- identify people from biometric signals (iris, fingerprint, signature, face image)
- detect people faces in images
- monitor urban zones and detect abnormal behaviors (surveillance)
- align medical images and estimate the boundary of organs
- detect and classify diseases from medical images
- classify land cover using satellite images
- improve the quality of degraded images

These problems are very different and they are solved by different methods. However, all methods share common steps: i) development of a model for the information to be retrieved, ii) fitting the model to the image or group of images and iii) evaluation of the model. These three steps can be performed in different ways as we shall see during this course.

There is not a unified theory for Vision or for the human perception as we have in the field of Mechanics with the Lagrangean framework or in Electromagnetism with the Maxwell equations. The best idea we can provide about this field is that we have many challenges (problems) and a wide variety of mathematical methods (tools). When we want to solve a new problem we can probably use several methods we know. Experience allows us to guess which ones will work best and which kind of changes will be needed.

Image processing and computer vision are strongly connected to several disciplines e.g., statistical modeling, optimization, pattern recognition and tracking. However, it is fair to say that many of the techniques used in image analysis and vision can be classified as statistical learning methods.

This course tries to identify some of the key methods in Computer Vision and Image Processing and provides a discussion of a few challenging problems hoping that this approach will provide a useful introduction to the field.

Sources

There are excellent textbooks in specific areas of Image Processing and Computer Vision e.g., 3D vision [1, 2], wavelets [3], image reconstruction [4]).

It is more difficult to suggest a single textbook with broad scope. The area is so vast that it is difficult to select the topics and to identify which are the most important. It all depends on the focus. There is also a difficult trade off between a large scope and depth. Some classic books which cover many aspects of image processing and computer vision are [6, 7, 8, 9, 10]. An good alternative for a broad introduction to this field is given by the *Handbook of Image and Video Processing* [11] written by several top researchers in this area. Electronic versions of several books are available in the CVOnline site (http://homepages.inf.ed.ac.uk /rbf/CVonline/books.htm).

Despite the effort to write good textbooks most of the information is available on scientific journals. A few examples are:

- Transactions Pattern Analysis and Machine Vision
- Transactions on Image Processing
- Transactions Medical Imaging
- International Journal of Computer Vision
- Image and Vision Computing

The tour is about to begin. I hope you like it.

Jorge S. Marques

Bibliography

[1] Y. Ma, S. Soatto, J. Kosecka, S. Sastry, An Invitation to 3D Vision: From Images to Geometric Models, Springer Verlag, 2003.

[2] R. I. Hartley, A. W. Zisserman, Multiple View Geometry in Computer Vision, Second Edition, Cambridge University Press, 2001.

[3] S. Mallat, A Wavelet Tour of Signal Processing, Academic Press, 2nd edition, 1999.

[4] H. C. Andrews, B. R. Hunt, Digital Image Restoration, Prentice Hall, 1977

[5] A. Blake, A. Zisserman, Visual reconstruction , MIT Press, 1987

[6] D. Forsyth and J. Ponce, Computer Vision a Modern Approach (Support Website), Prentice Hall, 2003, ISBN 0-13-085198-1.

[7] R.C. Gonzalez, R.E. Woods, Digital Image Processing (2nd edition), Prentice Hall, 2002.

[8] B.K.P. Horn, Robot Vision, McGraw Hill, 1986,

[9] M. Haralick, L.G. Shapiro, Computer and Robot Vision, Addison-Wesley, 1992.

[10] M. Petrou, P. Bosdogianni; Image Processing: The Fundamentals, John Wiley and Sons, 1999.

[11] A. Bovik, ed., Handbook of Image and Video Processing, Academic Press, 2000

Part I

Basics

Chapter 1

Images

1.1 What is an image?

Images are signals which represent the evolution of a quantity in the plane or in space.

Let us consider for example a gray level image. Each image point $\mathbf{x}$ is associated to a real number (intensity) $I(\mathbf{x})$ which measures the amount of radiation received at the vicinity of $\mathbf{x}$. The set of points for which the signal is defined in called the domain $\mathcal{D}$. A gray level image is a signal which associates a real value to each point $\mathbf{x} \in \mathcal{D}$.

Figure 1.1 shows a well known image called boat (left). The image can be visualized as an intensity map which associates an elevation to each point of the domain. The intensity map for a small block of the image boat is shown in figure 1.1 (center). Try to guess which region is represented without looking at the block on the top. It is not simple. We are not used to interpret such intensity surfaces. The intensity surface has the same information as the original image. However we are not able to analyse its content most of the time since our visual system is not trained for that task.

There are several types of images. Typical examples are color images. The color is a human perception, a response of the human brain to the radiation received at the retina cells. Fortunately, few information is extracted from electromagnetic spectrum by the retina and only three coefficients (e.g., the intensities of three primary colors RGB) are needed to specify the color perception. Color images are represented in a computer by the intensities of R (red), G (green) and B (blue) i.e. by a vector with three components. Therefore, a color image is a signal which associates each point of the domain with a vector belonging to $\mathbb{R}^3$, i.e., $I : \mathcal{D} \to \mathbb{R}^3$.

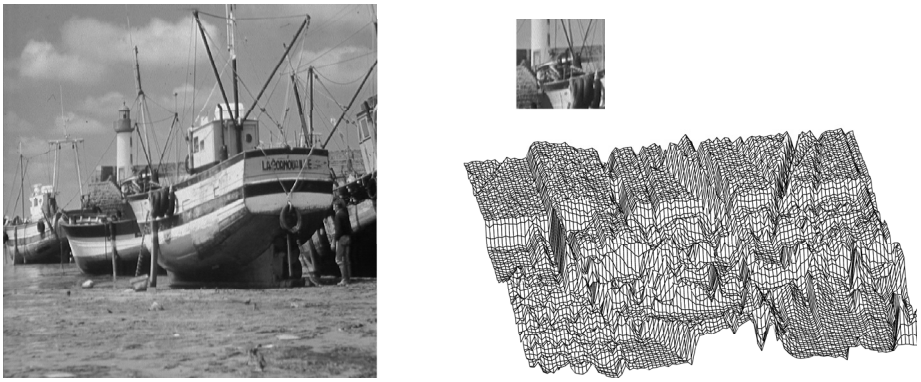


Figure 1.1: Image boat, intensity surface (block) and intensity profile along the main diagonal

From a mathematical point of view, an image is a map $I : \mathcal{D} \to \mathcal{R}$ where the domain $\mathcal{D}$ is a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$.

Examples

Two examples are the images defined in the interval of $\mathbb{R}^2$ e.g. in a domain $\mathcal{D} = [0, 1]^2$ or in a grid of points $\mathcal{D} = \{0, 1, \ldots, M-1\} \times \{0, 1, \ldots, N-1\}$

When the domain is $\mathbb{R}^2$ or an interval of $\mathbb{R}^2$ the image is denoted as a *continuous image*. When the domain is $\mathbb{Z}^2$ or an interval of $\mathbb{Z}^2$ the image is called a *discrete image*.

This raises a question: is it possible to store a continuous image since it has an infinite number of points? we will address this question later.

Range

The range of an image depends on the sensor used during the acquisition process and the transformations applied to the image. Examples:

- **real images** are produced by many sensors: CCD cameras, computed tomography, ultrasound sensors, etc. In this case the co-domain is $\mathcal{R} = \mathbb{R}$.
- **vector images** map each point of the domain to a vector of ($\mathcal{R} = \mathbb{R}^n$). The simplest case are the images produced by color cameras which associate a vector with three components

(RGB) to each point. Another example are satellite images which measure the intensity of the radiation in several frequency bands from the visible to the infra-red.

- **complex images** are produced by magnetic resonance imaging systems (MRI) and in many image processing operations (e.g., Fourier transform). They map each point of the domain to a complex number.

- **label images** associate a *label* to each point. The *label* can be represented by an integer number belonging to a set of admissible *labels* $\mathcal{R} = \{0, 1, ..., L-1\}$. The labels may identify different objects in the image (p.ex., vehicles, people, obstacles).

1.2 Discrete Images

Discrete images are images whose domain is $\mathbb{Z}^2$ or a subset of $\mathbb{Z}^2$. Most sensors produce finite discrete images with $M \times N$ elements. The elements are usually associated to a rectangular grid and can be stored in a matrix

$$\mathbf{I} = \begin{bmatrix} I(0,0) & I(0,1) & \dots & I(0,N-1) \\ I(1,0) & I(1,1) & \dots & I(1,N-1) \\ \dots & \dots & \dots & \dots \\ I(M-1,0) & I(M-1,1) & \dots & I(M-1,N-1) \end{bmatrix} \tag{1.1}$$

For the sake of simplicity we will assume that $M = N$.

It is also possible to represent the $M \times N$ image as a column vector with MN components. This can be done by stacking the columns of the $M \times N$ matrix into a single vector. This is called the *vect* operation.

$$\mathbf{i} = vect(\mathbf{I}) = \begin{bmatrix} I(0,0) \\ \vdots \\ I(M-1,0) \\ \vdots \\ \vdots \\ I(0,N-1) \\ \vdots \\ I(M-1,N-1) \end{bmatrix}^T \tag{1.2}$$

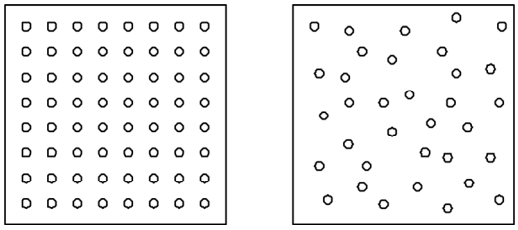


Figure 1.2: (a) uniform and (b) nonuniform sampling

We will adopt the following notation. Matrices and vector will be represented with bold letters. Matrices will be represented by capital letters.

Two simple operations we can define on the space of $M \times N$ images are the addition of two images and the multiplication of an image by a constant (scalar). These operations can be easily defined using matrix (left) and vector (right) notations

$$\begin{aligned} \mathbf{W} &= \mathbf{U} + \mathbf{V} & \mathbf{w} &= \mathbf{u} + \mathbf{v} \\ \mathbf{W} &= \alpha \mathbf{U} & \mathbf{w} &= \alpha \mathbf{u} \end{aligned} \tag{1.3}$$

The set of $M \times N$ images with these two operations is a vector space of dimension MN.

1.3 Continuous Images

Continuous images are defined in an infinite set of points. *Is it possible to store continuous images in a computer?*

At a first glance the answer is negative. A continuous image is defined in an infinite number of points and therefore requires an infinite amount of information. This difficulty can be overcome if we impose additional restrictions e.g., assuming that the image is an interpolated version of a finite set of samples (see figure 1.2a). The problem can be stated as follows.

Problem: given $I(k,l), k,l = 0, ..., M-1$ we wish to compute $I(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^2$.

This is known as the interpolation problem [2]. The interpolation is a key step in many image processing algorithms which require resampling the discrete image at different locations (e.g., rotations, translations with sub pixel resolution). These operations are used in many applications e.g., in medical imaging, remote sensing, surveillance, 3D vision, etc.

The classic solution consists of approximating the continuous function as a sum of interpolation functions centered at the nodes of a regular grid, multiplied by the image value at those points. Therefore,

$$I(\mathbf{x}) = \sum_{\mathbf{k}\in\mathbb{Z}^2} I(\mathbf{k})\phi(\mathbf{x}-\mathbf{k}) \tag{1.4}$$

where $\mathbf{k} \in \mathbb{Z}^2$ and ϕ is an interpolation function defined in $\mathbb{R}^2$.

The interpolated function $I(\mathbf{x})$ should match the samples values at the nodes of the interpolation grid, $I(\mathbf{k})$. Therefore, the following property applies

$$\phi(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{0} \\ 0 & \text{if } \mathbf{x} \in \mathbb{Z}^2 \setminus \{\mathbf{0}\} \end{cases} \tag{1.5}$$

where $\mathbf{0} = (0, 0)$ is the origin. It is also useful to use separable functions

$$\phi(\mathbf{x}) = \phi_0(x_1)\phi_0(x_2) \tag{1.6}$$

where $\mathbf{x} = (x_1, x_2)$.

Examples

Some interpolation functions defined in $\mathbb{R}$ are

$$\phi(x) = \begin{cases} 1 & \text{if } |x| < 0.5 \\ 0.5 & \text{if } |x| = 0.5 \\ 0 & \text{c.c.} \end{cases} \qquad \phi(x) = \begin{cases} 1-|x| & \text{if } |x| < 1 \\ 0 & \text{c.c.} \end{cases} \qquad \phi(x) = \frac{sin(\pi x)}{\pi x} \tag{1.7}$$

These functions verify condition (1.5). This condition is not unique. We can impose additional conditions. For example, given a constant discrete image, $I(\mathbf{k}) = 1$, the interpolated function should also be constant $I(\mathbf{x}) = 1, \forall \mathbf{x} \in \mathbb{R}^2$.

Using (1.4) we obtain

$$\sum_{\mathbf{k}} \phi(\mathbf{x}-\mathbf{k}) = 1 \qquad \forall \mathbf{x} \in \mathbb{R}^2 \tag{1.8}$$

A set of functions ϕ verifying this condition is called a partition of the unity.

The interpolation of a discrete image defined on a regular grid can be performed in a different way assuming that $I(\mathbf{x})$ is the sum of base functions $\psi(\mathbf{x})$ centered at the nodes of a regular

grid, multiplied by appropriate constants

$$I(\mathbf{x}) = \sum_{k \in \mathbb{Z}^2} c(\mathbf{k}) \psi(\mathbf{x} - \mathbf{k}) \tag{1.9}$$

The difference with respect to the previous approach lies on the fact that condition (1.5) may not be valid since we do not require that $c(\mathbf{k}) = I(\mathbf{k})$. This approach is known as the generalized interpolation problem and allows a greater flexibility in the choice of the basis functions. A popular choice for ψ are the B-spline polynomials.

The computation of the coefficients $c(\mathbf{k})$ involves the solution of a linear system of equations or filtering the discrete signal $I(\mathbf{k})$ using an appropriate filter. In the case of the B-splines, these operations can be done in an exact way using the algorithms described in [1].

A more general question can now be asked: *can we still reconstruct* $I(\mathbf{x})$ *when the samples are obtained at nonuniform positions?*

This is a more difficult problem but it still has practical interest. One of the motivations for this problem is in scope of data compression since we may wish to represent an image by a smaller set of chosen samples.

Most of the information conveyed by an image is located in regions where intensity rapidly changes (e.g., close to the edges). To represent an image by a set of points, the sampling density should be high at the transitions and sparse at homogeneous regions. The nonuniform interpolation algorithms are very useful since they allow to recover the original image from a small fraction of its elements.

An interpolation strategy consists of making $I(\mathbf{x})$ equal to the nearest sample (nearest neighbor method) in the image plane $\mathbf{x}_{(1)}$

$$I(\mathbf{x}) = I(\mathbf{x}_{(1)}) \tag{1.10}$$

This is a simple idea. However, the interpolated image is discontinuous and its intensity at location $\mathbf{x}$ depends on a single sample.

There are better ways to solve this problem e.g., assuming that the image is band limited non uniform reconstruction algorithms can be used. Another alternative consists of approximating the image by the superposition of spline basis functions located at the nodes of a regular grid [3], or splitting with polygonal meshes. This is an active research topic.

1.4 Color images

Many images obtained with CCD cameras are color images. This raises a question which we should address:

what is color and how do we represent color images in a computer?

Color is a human perception. It is the output of the human visual system to electromagnetic waves (light) arriving at the retina cells. In the 19th century, Young and Helmholtz found that it is possible to synthesize most colors by mixing three primary colors. This is a remarkable discovery since it allows us to display color in a computer or in a TV.

Suppose we receive a light ray at the retina with spectrum $S(\lambda)$ where λ is the wavelength and suppose that the three primary colors have known spectra $P_j(\lambda), j = 1, 2, 3$. In general, $S(\lambda)$ is a complex function and cannot be represented by a linear combination of three spectra $P_j(\lambda)$ i.e.,

$$S(\lambda) \neq \sum_{j=1}^{3} c_j P_j(\lambda) \tag{1.11}$$

where c_j are the mixing coefficients.

How do we combine this fact with Young and Helmoltz discovery? Three primary colors are not enough to accurately synthesize the input spectrum $S(\lambda)$. The amazing thing is that, despite this difference, the color perception can be the same.

To understand this we must understand how retina works. Retina has three different types of cells sensitive to color, denoted as cones. Two spectra may be different and still produce the same response in the three types of cones.

The output of the cone $i \in \{1, 2, 3\}$ to the incident spectrum can be modeled as

$$r_i = \int S(\lambda) S_i(\lambda) d\lambda \tag{1.12}$$

where $S_i(\lambda)$ is the sensitivity curve of cone i. The sensitive curve has an intuitive meaning: $S_i(\lambda_0)$ is the output of cone i when it is exposed to a monochromatic wave with spectrum $\delta(\lambda - \lambda_0)$.

Using (1.12), we conclude that $S(\lambda)$ and $\sum c_j P_j(\lambda)$ represent the same color iif[1]

$$\int S(\lambda) S_i(\lambda) d\lambda = \int \sum_j c_j P_j(\lambda) S_i(\lambda) d\lambda \qquad i = 1, 2, 3 \tag{1.13}$$

$$\int S(\lambda) S_i(\lambda) d\lambda = \sum_j c_j \int P_j(\lambda) S_i(\lambda) d\lambda \qquad i = 1, 2, 3 \tag{1.14}$$

[1] *iif* means *if and only if*

This is a system of three linear equations. If we define,

$$R_{ij} = \int P_j(\lambda) S_i(\lambda) d\lambda \tag{1.15}$$

$$r_i = \int S(\lambda) S_i(\lambda) d\lambda \tag{1.16}$$

then

$$\mathbf{Rc} = \mathbf{r} \tag{1.17}$$

where $\mathbf{R} = [R_{ij}], \mathbf{r} = [r_i], \mathbf{c} = [c_i]$. These are the conditions which must be met in order to obtain the same color perception with three primary colors. The mixture is realizable if all the coefficients are non negative $c_i \geq 0$ (additive mixture). This is not true in some cases. When the solution of (1.17) leads to negative coefficients the color is not realizable by an additive mixture of three primary colors$P_k(\lambda)$[2]. *Can three primary spectra $P_k(\lambda)$ synthesize all the colors with positive coefficients?* the answer is *no.*

Color can be represented by several systems of coordinates. The most popular is the RGB system which is based on three monochromatic colors with wavelengths $700nm$ (red), $546.1nm$ (green) and $435.8nm$ (blue). Each color is represented in this system by three coordinates which are the mixture coefficients required to synthesize the color. Other systems of coordinates such as XYZ, YUV, HSV (hue, saturation, value) and CMY used for printing.

A color image I is a signal which assigns a 3 color coordinates to each point $x \in \mathcal{D}$. Each value is therefore a 3-vector. As alternative we can think of a color image as three gray level images representing the color coordinates. Figure 1.3 shows an example of a color image and the three color components in the RGB system (2nd line) and HSV system.

Each coordinate conveys useful information. For example, the hue is often used to discriminate colors and to detect the skin region in an image.

[2] adding more primary colors might help in this case

Figure 1.3: Color image and color coordinates in RGB and HSV systems

Bibliography

[1] M. Unser, B-Splines: a Perfect Fit for Signal and Image Processing, IEEE Signal Processing Magazine, 23-37, November 1999.

[2] P. Thévenaz, T.Blu, M. Unser, Image Interpolation and Resampling, Handbook of Medical Imaging, Processing and Analysis, I.N. Bankman, Ed., Academic Press, 393-420, 2000.

[3] M. Arigovindan, M. Sühling, P. Hunziker, M. Unser,Variational Image Reconstruction From Arbitrarily Spaced Samples: A Fast Multiresolution Spline Solution, IEEE Transactions on Image Processing, Vol. 14, No. 4, April 2005.

Chapter 2

Filtering

2.1 Introduction

Image filtering aims to eliminate undesired structures from the image and enhance others. Filtering is used to smooth images, to reduce noise or to enhance intensity transitions or corners. We can design filters to enhance vertical or horizontal transitions, squares, circles or to interpolate the image when a few samples are not observed. Figure 2.1 shows a couple of examples.

In all these cases, the filter maps an input image I into a modified image J. The map can be briefly stated as follows

$$J = T(I) \tag{2.1}$$

There are several types of filters. This sections considers three classes: linear filters, order filters and morphological filters. Linear filters are very well studied from a theoretical point of view since there is a comprehensive theory to describe their behavior and stability. However, non linear filters often achieve better results.

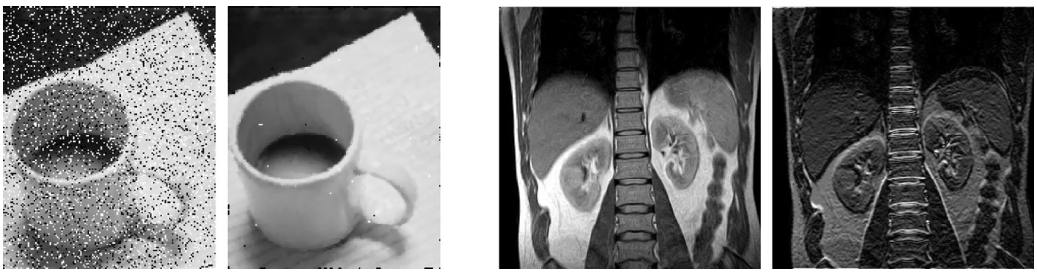


Figure 2.1: Filtering examples: input images (left) and filtered images (right)

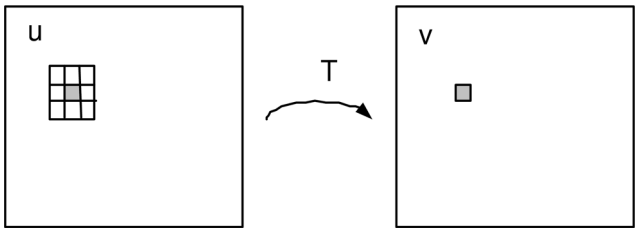


Figure 2.2: Linear filtering

2.2 Linear Filtering

This section addresses linear space invariant (LSI) filters. A simple example is a filter which computes a local average of the input image in a 3×3 neighborhood of each pixel (see Fig. 2.2). This operation involves multiplying the input image by a group of 9 coefficients which is often denoted as *mask*. The coefficients are moved along the input image as shown in the figure.

Let us now define two key concepts for this discussion: *linearity and space invariance.*

Definition (linearity): A filter T is linear if it obeys the superposition principle

$$T(aI_1 + bI_2) = aT(J_1) + bT(J_2) \tag{2.2}$$

for all input images I_1, I_2 and constants a, b.

This has an intuitive meaning. We can either filter a linear combination of two images or filter each them separately and combine the outputs. This property strongly restricts the class of transformations allowed. It is interesting to note that a constant map $T(I) = C$ where C is a constant image is not linear (unless $C = 0$).

Linearity is very useful when we decompose the image as a linear combination of basis images since we can express the output of the filter as a linear combination of the filtered basis images. Specifically, if

$$I(\mathbf{k}) = \sum_i c_i \phi_i(\mathbf{k}) \tag{2.3}$$

then

$$v(\mathbf{k}) = \sum_i c_i \psi_(\mathbf{k}) \tag{2.4}$$

where $\psi_i = T(\phi_i)$. The coefficients remain the same[1].

The second restriction comes from space invariance. A space invariant filter is a filter such that if the input image is translated by an amount $\mathbf{d} \in \mathbb{R}^2$, the output is also translated by $\mathbf{d}$.

[1]For the sake of simplicity we will assume that all the images are discrete images in this chapter.

Definition (space invariance): A filter is space invariant if

$$T(I(\mathbf{p}-\mathbf{d})) = J(\mathbf{p}-\mathbf{d}) \tag{2.5}$$

for all $J = T(I)$ and displacement $\mathbf{d} \in \mathbb{R}^2$.

A filter is linear and space invariant (LSI) if both properties hold.

We can now state the most important result for LSI filters. If a filter is linear and space invariant, its output is given by the convolution sum

$$J(\mathbf{k}) = \sum_{\mathbf{i}\in\mathbb{Z}^2} I(\mathbf{i})H(\mathbf{k}-\mathbf{i}) \tag{2.6}$$

where $u(\mathbf{k})$ is the input image ans $h(\mathbf{k})$ is the response of the filter to a an impulse image $\delta(\mathbf{k})$ ($\delta(\mathbf{k}) = 1$ if $\mathbf{k} = (0,0)$, $\delta(\mathbf{k}) = 0$ otherwise). The convolution sum is often denoted by

$$J = I * H \tag{2.7}$$

The proof is very simple. It is based on the representation of I as a sum of impulses (please convince yourself that this is true)

$$I(\mathbf{k}) = \sum_{\mathbf{i}\in\mathbb{Z}^2} I(\mathbf{i})\delta(\mathbf{k}-\mathbf{i}) \tag{2.8}$$

Applying the filter operator T to both members of this equation

$$v(\mathbf{k}) = T(u(\mathbf{k})) = T\left(\sum_{\mathbf{i}\in\mathbb{Z}^2} I(\mathbf{i})\delta(\mathbf{k}-\mathbf{i})\right) \tag{2.9}$$

$$v(\mathbf{k}) = \sum_{\mathbf{i}\in\mathbb{Z}^2} I(\mathbf{i})\; T\left(\delta(\mathbf{k}-\mathbf{i})\right) \qquad \text{linearity} \tag{2.10}$$

$$v(\mathbf{k}) = \sum_{\mathbf{i}\in\mathbb{Z}^2} I(\mathbf{i})\; h(\mathbf{k}-\mathbf{i}) \qquad \text{space invariance} \tag{2.11}$$

as wished.

This result is remarkable. It states that the filter behavior is completely characterized by the response of the filter to an impulse input. To design a LSI filter all we have to do is to define the impulse response $H(\mathbf{k})$.

It is more intuitive to define a set of coefficients (mask) $M(\mathbf{k})$ which is moved along the image and multiply the image at each position (see Figure 2.2). The filter mask is the impulse response after an horizontal and vertical flip,

$$M(m,n) = H(-m,-n) \tag{2.12}$$

The convolution sum can now be rewritten as follows

$$J(\mathbf{k}) = \sum_{\mathbf{i}\in\mathbb{Z}^2} I(\mathbf{i})M(\mathbf{k}-\mathbf{i}) \tag{2.13}$$

This equation is more intuitive. The filter output is obtained by sliding the mask along the input image. Then both images are multiplied and added. Many filtering operations are done in this way using (2.13).

Example

Consider filter with impulse response H, mask M and an input image I (the origin of each image is displayed in bold)

$$H = \begin{bmatrix} \mathbf{1} & -1 \\ 1 & 2 \end{bmatrix} \qquad M = \begin{bmatrix} 2 & 1 \\ -1 & \mathbf{1} \end{bmatrix} \qquad I = \begin{bmatrix} \mathbf{1} & 2 \\ 4 & 3 \end{bmatrix} \tag{2.14}$$

The filter output is

$$J = \begin{bmatrix} \mathbf{1} & 1 & -2 \\ 5 & \mathbf{3} & 1 \\ 4 & 11 & 6 \end{bmatrix} \tag{2.15}$$

The properties of LSI filters are well studied. Stability is one of the most important ones. A filter is stable if for every bounded input the output is also bounded. It can be easily proved that a LSI filter is stable if (and only if) the impulse response is absolutely summable i.e.

$$\sum_{\mathbf{k}\in\mathbb{Z}^2} |h(\mathbf{k})| < \infty \tag{2.16}$$

All the filters with finite impulse response are stable. Only filters with infinite impulse response may be unstable. For example, a filter with a constant impulse response $h(\mathbf{k}) = 1$ is unstable.

One way to define a filters with infinite impulse response is by a recursive equations e.g.,

$$J(m,n) = b_{00}I(m,n) + a_{10} * J(m-1,n) + a_{01} * J(m,n-1) \tag{2.17}$$

The equation allows us to filter the input image I with a small number of operations per sample.

Examples of linear filtering are:

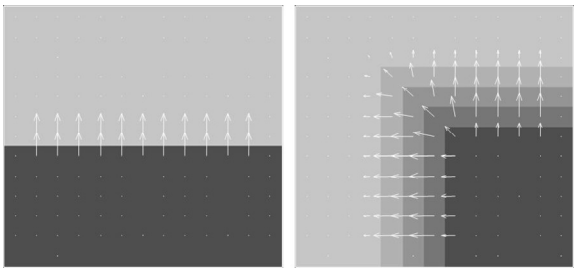


Figure 2.3: Gradient vector field

Image smoothing: a smoothing effect (low pass filtering) can be obtained by computing a weighted average of the input pixels. The mask coefficients should be positive and sum 1. A typical choice is the Gaussian mask

$$M(\mathbf{k}) = Ce^{-\frac{|\mathbf{k}|^2}{2\sigma^2}} \tag{2.18}$$

$\mathbf{k} = (m, n)$, $m, n \in \{-M, ..., M\}$ **Partial derivatives:** Partial derivatives can only be defined in an exact way for continuous images. In the case of discrete images, we can only compute approximate values but they are still very useful. Partial derivatives are usually approximated by filtering operations

$$\frac{\partial I}{\partial x_1} \approx H_1 * I \qquad \frac{\partial I}{\partial x_2} \approx H_2 * I \tag{2.19}$$

using appropriate masks M_1, M_2 e.g., the Sobel masks

$$M_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad M_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.20}$$

Image gradient: The image gradient is a 2D vector of partial derivatives

$$\frac{dI}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial I}{\partial x_1} \\ \frac{\partial I}{\partial x_2} \end{bmatrix} \tag{2.21}$$

which can be computed using the Sobel masks for example. The norm of the gradient and its direction convey useful information about abrupt changes of intensity which can be used for edge and corner detection. Figure 2.2 shows the gradient computed for two synthetic images using the Sobel masks. It displays the original image and the gradient vector at each point. The gradient has a large magnitude close to the transitions and it is orthogonal to the level curves of the intensity function.

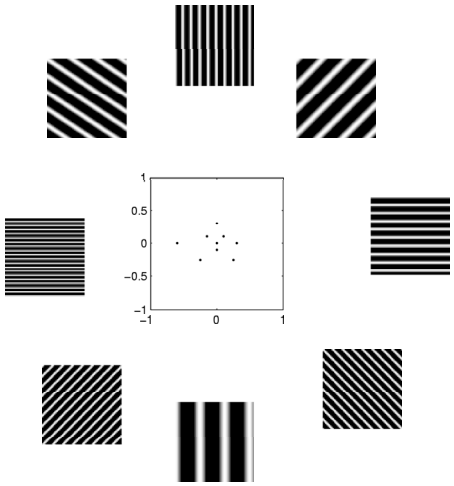


Figure 2.4: Real part of complex exponentials

2.3 Frequency Analysis

Frequency analysis is a useful tool for the design of linear filters.

It is based on the representation of images as a sum of complex exponentials

$$e^{j\omega\cdot\mathbf{k}} = e^{j(\omega_1 m+\omega_2 n)} \tag{2.22}$$

where $\omega = (\omega_1, \omega_2)$ is a vector of frequencies along the coordinate axis. The complex exponentials are periodic in ω_1, ω_2 with period 2π.

$$e^{j(\omega+2\pi\mathbf{i})\cdot\mathbf{k}} = e^{j\omega\cdot\mathbf{k}} \qquad \forall\mathbf{i}\in\mathbb{Z}^2 \tag{2.23}$$

This means that we do not have to consider the whole frequency plane but only an interval $[-\pi,\pi[^2$ since all exponentials are identical to one of these. Figure 2.3 shows the real part of complex exponentials for different values of $\omega \in [-\pi,\pi[^2$: $\|\omega\|$ controls the period of the waveform and the direction of ω defines the direction of the periodic pattern.

The Fourier transform of a discrete image and the inverse transform are defines as follows:

$$\tilde{I}(\omega) = \sum_{\mathbf{k}\in\mathbb{Z}^2} I(\mathbf{k})e^{-j\omega\cdot\mathbf{k}} \qquad \text{Fourier transform} \tag{2.24}$$

$$I(\mathbf{k}) = \frac{1}{4\pi^2}\int_{[-\pi,\pi]^2} \tilde{I}(\omega)e^{j\omega\cdot\mathbf{k}}d\omega \qquad \text{inverse Fourier transform} \tag{2.25}$$

Figure 2.5: Fourier transform of 8 images: image (left) and magnitude of the Fourier transform (right))

where $\tilde{I}(\omega)$ is the spectrum of the image $I(\mathbf{k})$. The inverse transform states that the image can be synthesized by "adding" complex exponentials with infinitesimal amplitudes $\tilde{I}(\omega)d\omega$ (synthesis equation).

The spectrum is a complex function and periodic with period 2π as a function of ω_1 and ω_2. It can be characterized by its modulo $|I(\omega)|$ and phase $\arg I(\omega)$.

Figure 2.3 shows some images (left) and the magnitude of the Fourier spectrum in the interval $[-\pi, \pi]$. The spectrum of sinusoids with frequency $\omega_0 \in [-\pi, \pi]^2$ are impulses in the frequency plane located at ω_0. When ω increases the frequency impulses move apart. Furthermore, if the image undergoes a rotation of amplitude θ the same happens to the spectrum. The second line shows rotated versions of a Gaussian function (Gabor Functions). The spectrum belongs to the same family. Is can be noted the effect of scale changes. A compression in space brings a dilation in the frequency domain and vice-versa.

The usefulness of this transform comes from the following properties:

- The response of a LSI filter to a complex exponential $e^{j\omega\cdot\mathbf{k}}$ is a complex exponential with the same frequency and different amplitude

$$\tilde{H}(\omega)e^{j\omega} \tag{2.26}$$

  where $\tilde{H}(\omega)$ is the Fourier transform of the filter impulse response and it is called the *Frequency Response* of the filter and the Fourier transform (analysis equation) tells us how to compute the coefficient of each exponential.

  Complex exponentials are very special signals since they are not modified by LSI filter T, apart from the amplitude. They are called eigen functions of the linear filters. The Fourier transform is therefore an expansion of the input signal as a sum of the filter

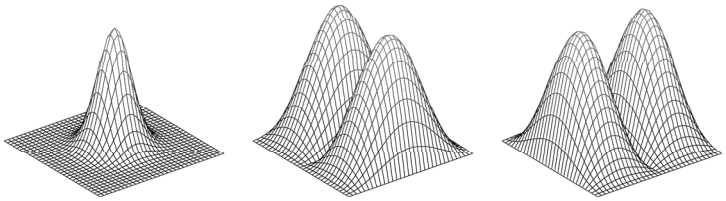


Figure 2.6: Frequency responses of Gaussian and Sobel filters as a function of ω_1, ω_2

eigenfunctions.

- The Fourier transform of the filter output is given by the multiplication of two spectra

$$\tilde{J}(\omega) = \tilde{H}(\omega)\tilde{I}(\omega) \tag{2.27}$$

The Fourier transform of the output image $\tilde{J}(\omega)$ is the product of the input Fourier transform $\tilde{I}(\omega)$ by the frequency response of the filter $H(\mathbf{k}), I(\mathbf{k})$.

The second property extends the same idea for multiple exponentials. The input can be decomposed as a sum of complex exponentials by the Fourier transform. The amplitude of each of exponential is multiplied by the filter frequency response $\tilde{H}(\omega)$.

If $|\tilde{H}(\omega)|$ is larger than 1 the exponential is amplified, otherwise it is attenuated. We can classify filters are low pass, high pass, bandpass, band reject, according to the range of frequencies they attenuate.

Fig 2.3 shows the amplitude of the frequency response for the Gaussian and Sobel filters. The frequency response of the Sobel filter is

$$\tilde{H}_1(\omega) = e^{-j(-\omega_1-\omega_2)} + 2e^{-j(-\omega_1)} + e^{-j(-\omega_1+\omega_2)} - e^{-j(-\omega_1-\omega_2)} - 2e^{-j(-\omega_1)} - e^{-j(-\omega_1+\omega_2)} \tag{2.28}$$

$$= (e^{j\omega_1} - e^{-j\omega_1})(e^{j\omega_2} + 2 + e^{-j\omega_2}) = 4j \sin\omega_1(1 + \cos\omega_2) \tag{2.29}$$

The frequency response can be used to understand the filter behaviors in the frequency domain and can also be used for filter design if we know the ideal frequency response for the problem.

2.4 Matrix Notation

We can represent finite signals and images by vectors. Linear filtering operations can therefore be expressed as matrix multiplications.

Let us consider the 1D signals and filter first. The convolution of two 1D signals $v = u * h$ of finite length n, m, can be written as follows

$$\mathbf{v} = \mathbf{H}\mathbf{u} \tag{2.30}$$

where $\mathbf{u}, \mathbf{v}$ are column vectors and $\mathbf{H}$ is a sparse $(n + m - 1) \times n$ matrix

$$\mathbf{H} = \begin{bmatrix} h(0) & & & & & \\ h(1) & h(0) & & & 0 & \\ \vdots & \vdots & \cdots & & & \\ h(m-1) & \cdots & h(1) & h(0) & & \\ & & & & & \\ & & & & h(1) & h(0) \\ & 0 & & & h(2) & h(1) \\ & & & & & \vdots \\ & & & & & h(m-1) \end{bmatrix} \tag{2.31}$$

Matrix $\mathbf{H}$ is a Toeplitz matrix since the elements of all diagonals are equal: $h(i, j) = h(i - j)$. The first column of $\mathbf{H}$ is the filter impulse response.

Let us consider now image filtering. The convolution of two finite support images $V = H * U$ with sizes $M \times M, N \times N$ can also be expressed in the same way

$$\mathbf{v} = \mathbf{H}\mathbf{u} \tag{2.32}$$

where $\mathbf{u}, \mathbf{v}$ are again column vectors with all the image elements. However, the structure of $\mathbf{H}$ is more complex. $\mathbf{H}$ it is a block matrix; each block $\mathbf{H}_{ij}$ represents the influence of the j-th column of the input image on the i-th column of the output. After some manipulations we

conclude that

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0 & & & & & & \\ \mathbf{H}_1 & \mathbf{H}_0 & & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & & \\ \mathbf{H}_m & \cdots & \cdots & \mathbf{H}_1 & \mathbf{H}_0 & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & & \\ & & & & & \mathbf{H}_1 & \mathbf{H}_0 \\ & & & & & \mathbf{H}_2 & \mathbf{H}_1 \\ \cdots & \cdots & \cdots & & & \cdots & \\ & & & & & \mathbf{H}_{m-1} & \mathbf{H}_m \end{bmatrix} \tag{2.33}$$

where

$$\mathbf{H}_k = \begin{bmatrix} h(0,k) & & & & & & \\ h(1,k) & h(0,k) & & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & & \\ h(m,k) & \cdots & \cdots & h(1,k) & h(0,k) & & \\ & & & & & & \\ & & & & & h(0,k) & \\ & & & & & h(1,k) & \\ \cdots & \cdots & \cdots & & & \cdots & \\ & & & & & h(m,k) & \end{bmatrix} \tag{2.34}$$

$\mathbf{H}$ is therefore a block Toeplitz matrix and each block is also a Toeplitz matrix.

This result can be derived as follows. The column i of the filtered convolution the pixels of the i-th column are given by

$$V(m,i) = \sum_p \sum_q H(m-p, i-q)U(p,q) \tag{2.35}$$

Let us consider the terms which depend on the j-th column of the input $q = j$ and forget the others for the moment

$$V(m,i) = \sum_p H(m-p, i-j)U(p,j) + \text{other} \tag{2.36}$$

This is the expression of a 1D convolution of $H(m, i-j) * U_j$. Therefore the block $\mathbf{H}_{ij}$ is a Toeplitz matrix whose first column is the impulse response $h(m, i-j), m = 0, \cdots, m-1$ as

we wished.

Example

The convolution of Example 2.14 can be formulated using matrix operations

$$\begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_0 & 0 \\ \mathbf{H}_1 & \mathbf{H}_0 \\ 0 & \mathbf{H}_1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} \qquad \mathbf{H}_0 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{H}_1 = \begin{bmatrix} -1 & 0 \\ 2 & -1 \\ 0 & 2 \end{bmatrix} \tag{2.37}$$

Although $\mathbf{H}$ is block Toeplitz it is not a Toeplitz matrix. Therefore the 2D convolution is not a 1D convolution.

2.5 Median Filter

The median filter is a nonlinear filter. It does not verify the linearity condition. It often performs better than linear filters, as we shall see, but it is harder to characterize its performance.

Median filter is based on the concept of order statistics. Let $u_1, u_2, \ldots, u_n$ be a set of iid random variables, with a common density $p(u)$ and distribution $P(u)$. The order statistics are obtained by arranging the n variables in ascending order of magnitude

$$u_{(1)} \leq u_{(2)} \leq \cdots \leq u_{(n)} \tag{2.38}$$

$u_{(k)}$ is called the $k-th$ order statistic.

The density of the $k-th$ order statistic is given by

$$p_k(u) = np(u) \begin{pmatrix} n-1 \\ k-1 \end{pmatrix} P(u)^{k-1}(1-P(x))^{n-k} \tag{2.39}$$

This can be proved as follows

$$\begin{aligned} p_k(u)du &= P\{u_{(k)} \in]u, u+du[\} && (2.40) \\ &= nP\{u(i) \in]u, u+du[\text{ and } (k-1) \text{ of the others smaller than u}\} && (2.41) \\ &= nP\{u(i) \in]u, u+du[\}\ P\{(k-1) \text{ of the others smaller than u}\} && (2.42) \\ &= np(u)du \begin{pmatrix} n-1 \\ k-1 \end{pmatrix} P(u)^{k-1}(1-P(x))^{n-k} && (2.43) \end{aligned}$$

Some of the order statistics $u_{(k)}$ are well known: $u_{(1)}$ is the minimum, $u_{(n)}$ is the maximum, and $u_{(\tau)}$, $(\tau = (n-1)/2,\ n \text{ odd})$ is the median. Consider for example a sequence

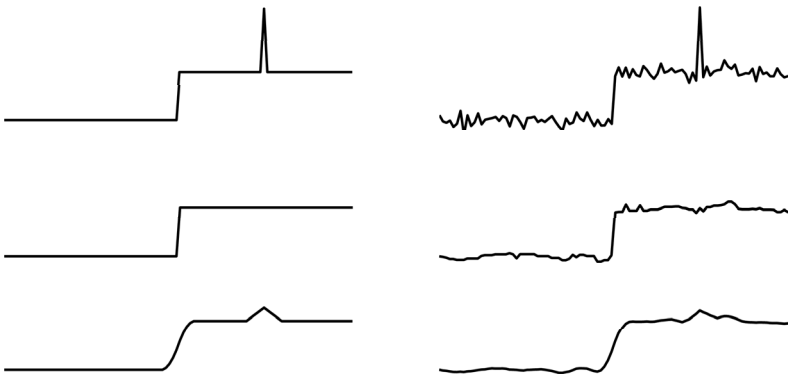


Figure 2.7: Median filtering: input (up), median filtering (middle) and linear filtering (down)

$\{0, 3, 12, 4, 1, 2, 2\}$. To compute the median, we order the 7 values $(0, 1, 2, 2, 3, 4, 12)$ and pick the fourth ($u_{med} = 2$).

The median filter does something similar. It computes the median of the input image in the vicinity of each point p

$$v(\mathbf{k}) = med\ \{u(q) : q \in V_p\} \tag{2.44}$$

where V_p is the vicinity of p e.g., defined by

$$V_p = \{q : |q - p| \leq r\} \tag{2.45}$$

This definition is valid for 1D and 2D signals.

The median filter rejects outliers and preserves discontinuities. Figure 2.5 shows an example. The first row displays two input signals: a step signal and an outlier, with and without Gaussian noise. The second row shows the output of the median filter of length 5. The median filter preserves the transitions and eliminates the outlier as we wish. The Gaussian noise is also attenuated. The third row displays the results of a linear filter with triangular impulse response of length 7. The linear filter attenuates the Gaussian noise even better but it smooths the discontinuity and does not eliminate the outlier.

The smoothing effect caused by the linear filter is a major drawback in image processing since transitions correspond to image edges and they are perceptually very important. Smoothed images are perceived as blurred and this may be more annoying than the Gaussian noise itself.

Another example of median filtering is shown in Fig. 2.1 (left) in which the input image is corrupted by "salt and pepper" noise with probability 0.1, i.e., 10% of the pixels were not observed and they are randomly replaced by 0 or 255. The performance of the median filter is very good in this problem.

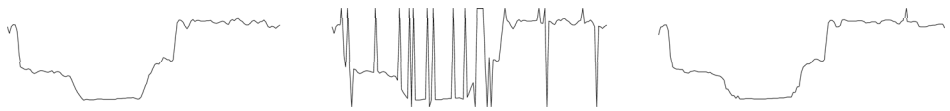


Figure 2.8: Median filtering: original signal (left), distorted signal with salt and pepper noise (center) and filtered signal (right)

The median value v minimizes the L_1 norm of the data

$$C = \sum_{q \in V} |u(q) - v| \tag{2.46}$$

This property is important because it allows several extensions of the median filter.

Median filtering of vector data

How can we apply median filtering to a color image? It is not a trivial question because it is not possible to order vectors in a meaningful way. We can apply the median filter to each component of the vector image but this is not a good idea. Instead, the median filter can be defined by the minimization of (2.46) where $u(q), v$ are vectors in this case. This minimization cannot be done analytically and have must use numerical optimization methods.

Weighted median filter

The median filter assigns an equal weight to all the data points, despite their position inside the window V_p. It is often useful to assign different weights to the input values according to their position. For examples, pixels close to the center of V_p are probably more important and should have a higher weight the pixels close to the boundary. Weighting can be done by modifying the cost function

$$C = \sum \sum_{q \in V} w(q) \, |u(q) - v| \tag{2.47}$$

If the weights are integers, the minimization of (2.47) is straightforward. All we have to do is to repeat each input data $w(q)$ times and compute the median of the extended sequence.

2.6 Order Filters

The median filter belongs to a wider class of order filters of the form

$$v(\mathbf{k}) = \sum_{k=1}^{n} a_k u_{(k)}(\mathbf{k}) \tag{2.48}$$

where $u_{(k)}$ is the kth order statistic of $\{u(q), q \in V_p\}$, $n = \#V_p$ and a_k are the filter coefficients.

A special case are the rank order filters

$$v(\mathbf{k}) = u_{(k)}(\mathbf{k}) \tag{2.49}$$

If $k = 1$ the rank order filter is called the minimum filter and if $k = n$ it is called the maximum filter.

Another order filter is the α-trimmed filter which discards a fraction α of the input samples and computes the average value of the others (we assume that $\alpha = j/n, j \in \mathbb{N}$)

$$v(\mathbf{k}) = \frac{1}{n(1-2\alpha)} \sum_{k=1+\alpha}^{n-\alpha n} u_{(k)} \tag{2.50}$$

The alpha trimmed filter discards the smallest and largest samples. It can therefore deal with outliers provided that their percentage is smaller than α. The α trimmed filter performs better than the median filter since it is able to reject the outliers and performs a better noise reduction.

Statistical formulation

The coefficients can also be obtained from a statistical formulation of the problem [1]. Suppose the signal is locally constant and corrupted by additive noise

$$u(k) = \theta + w(k) \tag{2.51}$$

where θ is a constant and $w(k)$ is a sequence of iid noise variables. We want to find the filter with minimum variance (to simplify the notation, we dropped the dependence on p)

$$V = E\{(\theta - v)^2\} \tag{2.52}$$

assuming that the filter is unbiased

$$E\{v\} = \theta \tag{2.53}$$

The bias condition can be written as

$$E\{\sum_{k=1}^{n} a_k(\theta + w_{(k)})\} = \theta \sum_{k=1}^{n} a_k + \sum_{k=1}^{n} a_k E\{w_{(k)}\} = \theta \tag{2.54}$$

Since the density of $w_{(k)}$ is symmetric, the optimal coefficients are symmetric ($a_k = a_{n+1-k}$) and $E\{w_{(k)}\} = -E\{w_{(n+1-k)}\}$. Therefore the previous equation can be simplified

$$\sum_{k=1}^{n} a_k = 1 \tag{2.55}$$

and the filter output can now be written as follows

$$v = \theta + a^T w \tag{2.56}$$

where $a = [a_1 \dots a_n]^T$ is the vector of coefficients and $w = [w_{(1)} \dots w_{(n)}]^T$ is the vector of noise order statistics. The variance is now given by

$$V = a^T R a \tag{2.57}$$

where $R = E\{ww^T\}$ is the correlation matrix of the noise order statistics. The minimization of V with the bias restriction can be obtained by minimizing the Lagrangean function

$$L = a^T R a - \lambda(e^T a - 1) \tag{2.58}$$

where $e = [1 \dots 1]^T$. Computing the derivative with respect to vector a we obtain

$$(R + R^T)a - \lambda e = 0 \qquad a = \frac{\lambda}{2} R^{-1} e \tag{2.59}$$

Since $e^T a = 1$, $\lambda = 2/(e^T R^{-1} e)$. The optimal coefficients are

$$a = \frac{R^{-1} e}{e^T R^{-1} e} \tag{2.60}$$

This method provides an optimal set of coefficients depending on the noise statistics. The only difficulty is the computation of the correlation entries $R_{ij} = E\{w_{(i)} w_{(j)}\}$

$$\mathbf{R}_{ij} = \int w_{(i)} w_{(j)} p(w_{(i)}, w_{(j)}) dw_{(i)} dw_{(j)} \tag{2.61}$$

There are analytic expressions for $p(w_{(i)} w_{(j)})$ but the expected value has to be computed by numerically integration in the plane. Recent works have try to overcome this difficulty [4].

Bibliography

[1] A. Bovik, T. Huang, D. Munson, A Generalization of Median Filtering Using Linear Combinations of Order Statistics, IEEE Trans. ASSP, 1342-1350, Dec. 1983.

[2] P. S. Huber, Robust Statistics, Wiley, 1981.

[3] I. Pitas, A. Venetsanopoulos, Order Statistics in Digital Image Processing, Proc. IEEE, Vol. 80, 1893-1921, Dec 1992.

[4] R. Öten, R. J. P. de Figueiredo, Sampled Function Weighted Order Filters, IEEE Trans. Circuits and Systems, Jan. 2002.

Chapter 3

Image Alignment

3.1 Introduction

We often have multiple images of the same object and wish to align them in order to compare them or to fuse them into a single image. This is what happen in a *mosaicing* task which aims to reconstruct a panoramic image (*mosaic*) from several partial views. Figure 3.1 shows an example in which two images are aligned to create a mosaic. This example is misleading. It is not possible to align images of 3D objects by a single transformation unless the objects are approximately planar or the camera motion is a pure rotation. This is what happens in this example.

Image alignment is also very important to compare medical images of the same patient obtained using different modalities (e.g., CT, MRI) at different instants of time. Alignment is also a key operation in surveillance or in stereo vision.

Our first attempt to formulate the problem is the following:

Problem 1: alignment of points

Given a set of points $\mathbf{x}_i$ *in an image* T *and a set of points* $\mathbf{y}_i$ *in an image* I *(see figure 3.2), we wish to find a geometric transformation mapping the points* $\mathbf{x}_i$ *into the points* $\mathbf{y}_i$*.*

Image T is called a template and the points $\mathbf{x}_i, \mathbf{y}_i$ are denoted as marks or fiducial points. Everything becomes more difficult if the matching between the two sets of points $\mathbf{x}_i, \mathbf{y}_i$ is unknown.

Alignment can also be done without marks. In this case we may assume that all the points

Figure 3.1: Image alignment: pair of images (1st row) and mosaic (2nd row)

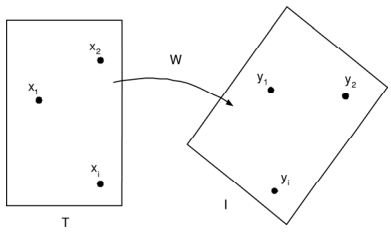


Figure 3.2: Point Alignment

of the template image T should be associated to points of I with similar color or intensity. This is called the color constancy hypothesis.

Problem 2: dense alignment

Given two images I, T we wish to find a geometric transformation $\mathbf{W}(\mathbf{x})$ which maps points from the template image T into points of I, such that $I(\mathbf{W}(\mathbf{x})) \cong T(\mathbf{x})$.

The alignment of isolated points approach will be discussed in the next sections. Dense alignment will be addressed later in this course.

3.2 Geometric Transformations

There are many geometric transformations which can be use to align pairs of images ranging from simple translation and rotations which preserve metric information (distances and angles) to more general transformations (e.g., projective transformations) which do not preserve metric information.

We will briefly review the most frequent transformations used in image alignment:

- Translation:

$$\mathbf{W}(\mathbf{x};\theta) = \begin{bmatrix} x + t_1 \\ y + t_2 \end{bmatrix} \tag{3.1}$$

  where $\theta = (t_1, t_2)^T$ is a translation vector. This transformation preserves distances, angles and parallelism of lines.

- Rigid body:

$$\mathbf{W}(\mathbf{x};\theta) = \mathbf{R}\mathbf{x} + \mathbf{t} \tag{3.2}$$

  $\theta = (\mathbf{R}, \mathbf{t})$ where $\mathbf{R} \in \mathbb{R}^{\mathbf{2\times 2}}$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^{\mathbf{2}}$ is a translation vector. Rotation matrices are unitary matrices ($\mathbf{R}^T\mathbf{R} = \mathbf{I}$) which do not perform axis inversions ($\det(\mathbf{R}) = 1$). The rigid body transformation performs a rotation followed by a translation. It preserves distances, angles and parallelism.

  The alignment of two or more sets of points using a rigid body transformation is known as Procrustes analysis.[1].

- Euclidean similarity:

$$\mathbf{W}(\mathbf{x};\theta) = s\mathbf{R}\mathbf{x} + \mathbf{t} \tag{3.3}$$

  $\theta = (s, \mathbf{R}, \mathbf{t})$ where s is a scale factor, $\mathbf{R} \in \mathbb{R}^{2\times 2}$ is a rotation matrix $\mathbf{t} \in \mathbb{R}^2$ a translation vector. The similarity transformation performs a rotation followed by a translation. It preserves angles and parallelism of lines.

- Affine transformation:

$$\mathbf{W}(\mathbf{x};\theta) = \mathbf{A}\mathbf{x} + \mathbf{t} \tag{3.4}$$

  $\theta = (\mathbf{A}, \mathbf{t})$ where $\mathbf{A} \in \mathbb{R}^{2\times 2}$ is a square nonsingular 2×2 matrix and $\mathbf{t} \in \mathbb{R}^2$ is a translation vector. This transformation performs rotation, translation, horizontal, vertical

[1] Procrustes was a bandit which offered a stay to tired travelers saying that he had a special bed which would fit to the traveler size. He did not explain the way he fitted the bed to the traveler. If the traveler was short he/she would be stretched until he/she fit. If it was tall he/she would be amputated.

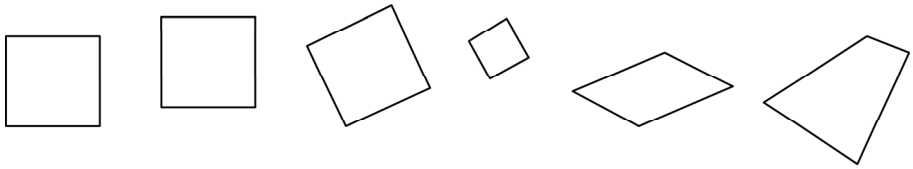


Figure 3.3: Geometric transformation of a square (left) with translation, rigid body, Euclidean similarity, affine transformation and projective transformation

and diagonal scaling. It preserves parallelism of lines but is does not preserve distances and angles.

- Projective transformation (homography):

$$\mathbf{W}(\mathbf{x};\theta) = \begin{bmatrix} \frac{\theta_1 x+\theta_2 y+\theta_3}{\theta_7 x+\theta_8 y+\theta_9} \\ \frac{\theta_4 x+\theta_5 y+\theta_6}{\theta_7 x+\theta_8 y+\theta_9} \end{bmatrix} \tag{3.5}$$

$\theta = (\theta_1, \ldots, \theta_9)$. This transformation maps straight lines into straight lines but does not preserve distances, angles or parallelism. We note that the vector θ is defined up to a non zero multiplicative factor. The transformation does not change if θ is multiplied by a constant $\mathbf{W}(\mathbf{x};\alpha\theta) = \mathbf{W}(\mathbf{x};\theta)$. This ambiguity consists can be solved by adding a additional constraint e.g., $||\theta|| = 1$ where $||.||$ stands for the Euclidean norm of a vector[2].

All the previous transformation map straight lines into straight lines. There are other more flexible transformations which map lines into curves.

3.3 Aligning sets of points

Let us assume that we want to align two sets of points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{2\times n}$. First we must choose the geometric transformation $\mathbf{W}(\mathbf{x},\theta)$. Then we need to estimate the alignment parameters θ. Let us address this step.

The main idea is the following. The parameters should be chosen in order to make the alignment errors in the image plane

$$\mathbf{e}_i = \mathbf{y}_i - \mathbf{W}(\mathbf{x}_i;\theta) \tag{3.6}$$

as small as possible.

[2]the Euclidean norm is defined by $||\mathbf{x}|| = \sqrt{\mathbf{x}^\mathbf{T}\mathbf{x}}$

In general we have more constraints than degrees of freedom and it is not possible to make all the errors equal to zero. Therefore, a fitness criterion is used to measure the amplitude of the alignment errors. The most popular choice is a quadratic criterion (error energy)

$$E(\theta) = \sum_i \|\mathbf{y}_i - \mathbf{W}(\mathbf{x}_i;\theta)\|^2 \tag{3.7}$$

which must be minimized with respect to θ.

This criterion can be written in a compact way if we define the data matrices $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_n], \mathbf{y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n], \mathbf{e} = [\mathbf{e}_1, \ldots, \mathbf{e}_n]$ with dimensions $2 \times n$. Since

$$\mathbf{e}\mathbf{e}^T = [\mathbf{e}_1, \ldots, \mathbf{e}_n] \begin{bmatrix} \mathbf{e}_1^T \\ \vdots \\ \mathbf{e}_n^T \end{bmatrix} = \begin{bmatrix} \sum_i e_{1i}^2 & \sum_i e_{1i}e_{2i} \\ \sum_i e_{2i}e_{1i} & \sum_i e_{2i}^2 \end{bmatrix} \quad , \tag{3.8}$$

the error energy is

$$E(\theta) = tr\{\mathbf{e}\mathbf{e}^T\} \tag{3.9}$$

where $tr\{.\}$ is the trace of a matrix (sum of the elements of the main diagonal). In the sequel, we will minimize $E(\theta)$ for each geometric transformation. We can use (3.7) or (3.9). We will use the second.

Translation

In this case $\mathbf{e} = \mathbf{y} - \mathbf{x} - \mathbf{t}\,\mathbf{1}^T$ ($\mathbf{1} \in \mathbb{R}^{n\times 1}$ is a vector of ones) and

$$E(\theta) = tr\{(\mathbf{y} - \mathbf{x} - \mathbf{t}\,\mathbf{1}^T)(\mathbf{y} - \mathbf{x} - \mathbf{t}\,\mathbf{1}^T)^T\} \tag{3.10}$$

$$E(\theta) = tr\{n\mathbf{t}\mathbf{t}^T - 2(\mathbf{y} - \mathbf{x})\mathbf{1}\mathbf{t}^T + (\mathbf{y} - \mathbf{x})(\mathbf{y} - \mathbf{x})^T\} \tag{3.11}$$

To minimize E we compute its derivative with respect to $\mathbf{t}$ and make it equal to zero (necessary condition). Then[3]

$$2n\mathbf{t} - 2(\mathbf{y} - \mathbf{x})\mathbf{1} = 0 \tag{3.12}$$

$$\mathbf{t} = \bar{\mathbf{y}} - \bar{\mathbf{x}} \tag{3.13}$$

where $\bar{\mathbf{x}} = \mathbf{x}\,\mathbf{1}/n, \bar{\mathbf{y}} = \mathbf{y}\,\mathbf{1}/n$. The optimal translation is

$$\mathbf{t} = \bar{\mathbf{y}} - \bar{\mathbf{x}} \tag{3.14}$$

where $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ are the mass centers of the two sets of points $\mathbf{x}_i$ e $\mathbf{y}_i$. The optimal translation shifts points $\mathbf{x}_i$ in order to align the mass centers of both sets of points.

[3]see the rules of the derivatives with respect to a matrix at the end of this report

Affine transform

In this case

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{t}^T\mathbf{1} \tag{3.15}$$

The motion parameters $\mathbf{A}, \mathbf{t}$ are estimated by minimizing (3.9). The estimation of $\mathbf{t}, \mathbf{A}$ can be carried out in two steps. The optimal translation is given by (prove it!)

$$\mathbf{t} = \bar{\mathbf{y}} - A\bar{\mathbf{x}} \tag{3.16}$$

The translation vector should align the mass centers. If we now subtract the mean vector from the data we obtain a new set of data points with zero mean $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ which are related by the model

$$\tilde{\mathbf{y}} = \mathbf{A}\tilde{\mathbf{x}} \tag{3.17}$$

Matrix $\mathbf{A}$ is estimated by minimizing the energy

$$E = tr\{(\tilde{\mathbf{y}} - \mathbf{A}\tilde{\mathbf{x}})(\tilde{\mathbf{y}} - \mathbf{A}\tilde{\mathbf{x}})^T\} = tr\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T - \tilde{\mathbf{y}}\tilde{\mathbf{x}}^T\mathbf{A}^T - \mathbf{A}\tilde{\mathbf{x}}\tilde{\mathbf{y}}^T + \mathbf{A}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\mathbf{A}^T\} \tag{3.18}$$

$$E = tr\{\mathbf{R}_{\tilde{y}\tilde{y}} - 2\mathbf{R}_{\tilde{y}\tilde{x}}A^T + A\mathbf{R}_{\tilde{x}\tilde{x}}A^T\} \tag{3.19}$$

where

$$\mathbf{R}_{\tilde{x}\tilde{x}} = \tilde{\mathbf{x}}\tilde{\mathbf{x}}^T = \begin{bmatrix} \sum \tilde{x}_{1i}^2 & \sum \tilde{x}_{1i}\tilde{x}_{2i} \\ \sum \tilde{x}_{2i}\tilde{x}_{1i} & \sum \tilde{x}_{2i}^2 \end{bmatrix} \qquad \mathbf{R}_{\tilde{y}\tilde{x}} = \tilde{\mathbf{y}}\tilde{\mathbf{x}}^T = \begin{bmatrix} \sum \tilde{y}_{1i}\tilde{x}_{1i} & \sum \tilde{y}_{1i}\tilde{x}_{2i} \\ \sum \tilde{y}_{2i}\tilde{x}_{1i} & \sum \tilde{y}_{2i}\tilde{x}_{2i} \end{bmatrix} \tag{3.20}$$

$$\mathbf{R}_{\tilde{y}\tilde{y}} = \tilde{y}\tilde{y}^T = \begin{bmatrix} \sum \tilde{y}_{1i}^2 & \sum \tilde{y}_{1i}\tilde{y}_{2i} \\ \sum \tilde{y}_{2i}\tilde{y}_{1i} & \sum \tilde{y}_{2i}^2 \end{bmatrix} \tag{3.21}$$

The derivative of E with respect to $\mathbf{A}$ can be easily obtained using the properties of the derivative of the trace with respect to a matrix see (10.33,10.37) in Appendix. Therefore,

$$-2\mathbf{R}_{\tilde{y}\tilde{\mathbf{x}}} + 2\mathbf{A}\mathbf{R}_{\tilde{\mathbf{x}}\tilde{\mathbf{x}}} = 0 \tag{3.22}$$

and

$$\mathbf{A} = \mathbf{R}_{\tilde{y}\tilde{\mathbf{x}}}\mathbf{R}_{\tilde{\mathbf{x}}\tilde{\mathbf{x}}}^{-1} \tag{3.23}$$

Equations (3.23,3.16) define the least squares estimate of the affine transform.

The algorithm can be summarised as follows.

Estimation of the Affine Transform

Given the data matrices $\mathbf{x}, \mathbf{y}$, with dimensions $2 \times n$:

- center the data at the origin: compute the mean value of the data $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ and subtract them from the columns of $\mathbf{x}$ e $\mathbf{y}$; The output is $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$.
- compute the covariance matrices $\mathbf{R}_{\tilde{x}\tilde{x}} = \tilde{\mathbf{x}}\tilde{\mathbf{x}}^T$, $\mathbf{R}_{\tilde{y}\tilde{x}} = \tilde{\mathbf{y}}\tilde{\mathbf{x}}^T$
- transform matrix $\mathbf{A}$: $\mathbf{A} = \mathbf{R}_{yx}\mathbf{R}_{xx}^{-1}$
- translation vector: $\mathbf{t} = \bar{\mathbf{y}} - \mathbf{A}\bar{\mathbf{x}}$

Rigid Body transformation*

The estimation of the rigid body transformation is known as procrustes analysis. In this case, matrix $\mathbf{A}$ is a rotation matrix. The minimization of E is more difficult since it is a constrained optimization problem. However, the final result is simple. It is based on the SVD decomposition of the cross covariance matrix $\mathbf{R}_{yx}$[4].

The rotation matrix $\mathbf{R}$ is a 2×2 matrix with a single degree of freedom (rotation angle). This happens because $\mathbf{R}$ has orthogonal columns with unit norm and determinant equal to 1. These constraints can be written as follows[5]

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} \qquad \det \mathbf{A} = 1 \tag{3.24}$$

We will use only the first restriction ($\mathbf{R}^T\mathbf{R} = I$) hoping that the second condition is enforced by the data.

The translation vector is estimated by aligning the mean vectors as before $\mathbf{t} = \bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}$. We can then subtract the mean vector to all the data and estimate the rotation matrix by minimizing (3.19) replacing $\mathbf{A}$ by $\mathbf{R}$.

$$E = tr\{\mathbf{R}_{\tilde{y}\tilde{y}} - 2\mathbf{R}_{\tilde{y}\tilde{x}}\mathbf{R}^T + \mathbf{R}\mathbf{R}_{\tilde{x}\tilde{x}}\mathbf{R}^T\} \tag{3.25}$$

The last term

$$tr\{\mathbf{R}\mathbf{R}_{xx}\mathbf{R}^T\} = tr\{\mathbf{R}^T\mathbf{R}\mathbf{R}_{xx}\} = tr\{\mathbf{R}_{xx}\} \tag{3.26}$$

[4] SVD stands for singular value decomposition of a matrix.

[5] a matrix $\mathbf{R} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ s called an orthogonal matrix if $\mathbf{R}^\mathbf{T}\mathbf{R} = \mathbf{R}\mathbf{R}^\mathbf{T} = \mathbf{I}$

does not depend on $\mathbf{R}$. Therefore, the minimization of E is equivalent to the maximization of

$$tr\{\mathbf{R}_{yx}\mathbf{R}\} \tag{3.27}$$

with the restriction $\mathbf{R}^T\mathbf{R} = \mathbf{I}$. This optimization problem can be solved by using the Lagrangean function

$$L(\mathbf{R}) = tr\{\mathbf{R}_{yx}\mathbf{R}\} + \Lambda(\mathbf{R}^T\mathbf{R} - I) \tag{3.28}$$

where Λ is a matrix of Lagrange multipliers. A necessary condition for optimality is

$$\frac{dL}{d\mathbf{R}} = \mathbf{0} \tag{3.29}$$

Therefore

$$\mathbf{R}_{yx} = \mathbf{\Lambda R} \tag{3.30}$$

where $\mathbf{R}$ is an orthogonal matrix. Therefore,

$$\mathbf{R}_{yx}\mathbf{R}_{yx}^T = \mathbf{\Lambda R R}^T\mathbf{\Lambda}^T = \mathbf{\Lambda\Lambda}^T \tag{3.31}$$

We will now use the SVD decomposition of $\mathbf{R}_{yx}$ given by

$$\mathbf{R}_{yx} = \mathbf{UDV}^T \tag{3.32}$$

where $\mathbf{U}, \mathbf{V}$ are orthogonal matrices and $\mathbf{D}$ is a diagonal matrix. Using (3.31) we obtain

$$\mathbf{\Lambda\Lambda}^T = \mathbf{UD}^2\mathbf{U}^T \tag{3.33}$$

We conclude that $\mathbf{\Lambda} = \mathbf{UDU}^T$. Replacing this in (3.30) we obtain

$$\mathbf{UDV}^T = \mathbf{UDU}^T\mathbf{R} \tag{3.34}$$

Therefore, $\mathbf{V}^T = \mathbf{U}^T\mathbf{R}$ and

$$\mathbf{R} = \mathbf{UV}^T \tag{3.35}$$

The rotation matrix it the product of two unitary matrices $\mathbf{UV}^T$ obtained from the SVD decomposition of $\mathbf{R}_{yx}$

Procrustes Analysis

Given the data matrices,$\mathbf{x}, \mathbf{y}$, with dimensions $2 \times n$:

- center the data at the origin: compute the average values of $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ and subtract them from the columns of $\mathbf{x}$ e $\mathbf{y}$; the output matrices are $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$.
- compute the cross covariance matrix $\mathbf{R}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}} = \tilde{\mathbf{y}}^T\tilde{\mathbf{x}}$ and its singular value decomposition $\mathbf{R}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}} = \mathbf{U}\mathbf{D}\mathbf{V}^T$
- rotation matrix: $\mathbf{R} = \mathbf{U}\mathbf{V}^T$
- translation vector: $\mathbf{t} = \bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}$

Projective Transform

The projective transform $\mathbf{W}(\mathbf{x};\mathbf{p})$ is a nonlinear function of $\mathbf{p}$. Therefore, the minimization of the energy (3.7) is a nonlinear optimization problem which does not have an analytical solution. If we want to minimize the energy we must resort to numerical optimization methods.

These difficulties can be overcome by changing the function to be minimized and converting it into a quadratic function of the parameters.

If we define $\mathbf{p}_1 = [p_1 p_2 p_3]^T, \mathbf{p}_2 = [p_4 p_5 p_6]^T, \mathbf{p}_3 = [p_7 p_8 p_9]^T$, e $\tilde{\mathbf{x}}_i = [\mathbf{x}_i\ 1^T]$, the projective transformation can be written as follows

$$y_{1i} = \frac{\tilde{\mathbf{x}}_i^T\mathbf{p}_1}{\tilde{\mathbf{x}}_i^T\mathbf{p}_3} \qquad y_{2i} = \frac{\tilde{\mathbf{x}}_i^T\mathbf{p}_2}{\tilde{\mathbf{x}}_i^T\mathbf{p}_3} \tag{3.36}$$

These equations allow us to define two algebraic errors (different from (3.6))

$$e_{1i} = \tilde{\mathbf{x}}_i^T\mathbf{p}_1 - (\tilde{\mathbf{x}}_i^T\mathbf{p}_3)y_{1i} \qquad e_{2i} = \tilde{\mathbf{x}}_i^T\mathbf{p}_2 - (\tilde{\mathbf{x}}_i^T\mathbf{p}_3)y_{2i} \tag{3.37}$$

and a quadratic cost function

$$C = \sum_{i=1}^{n} \|\mathbf{e}_i\|^2 \tag{3.38}$$

This cost function measures the amplitude of the algebraic errors but it has no geometric meaning. This error can be written using matrix notation

$$\mathbf{e} = \mathbf{M}\mathbf{p} \qquad \text{com} \qquad \mathbf{M} = \begin{bmatrix} \tilde{\mathbf{x}}_1^T & 0 & -\tilde{\mathbf{x}}_1^T y_{11} \\ 0 & \tilde{\mathbf{x}}_1^T & -\tilde{\mathbf{x}}_1^T y_{21} \\ \dots & \dots & \dots \\ \tilde{\mathbf{x}}_n^T & 0 & -\tilde{\mathbf{x}}_n^T y_{1n} \\ 0 & \tilde{\mathbf{x}}_n^T & \tilde{\mathbf{x}}_n^T y_{2n} \end{bmatrix} \tag{3.39}$$

Therefore $C = \theta^T \mathbf{M}^T \mathbf{M} \theta$. Since θ is to be estimated up to a multiplicative factor, we can add an additional constraint $\theta^T \theta = 1$. The minimization of C with this constraint can be done using the Lagrangean function

$$L = \theta^T \mathbf{M}^T \mathbf{M} \theta - \lambda(\theta^T \theta - 1) \tag{3.40}$$

where λ is a Lagrange multiplier.

A necessary condition to obtain a maximum of L at θ is obtained by making the derivative of L with respect to θ equal to zero

$$2\mathbf{M}^T \mathbf{M} \theta - 2\lambda\theta = \mathbf{0} \tag{3.41}$$

$$\mathbf{M}^T \mathbf{M} \theta = \lambda\theta \tag{3.42}$$

Therefore, the estimate of θ must be an eigenvector of $\mathbf{M}^T\mathbf{M}$.

There are 9 eigenvectors for $\mathbf{M}^T\mathbf{M}$. Which one should we choose. It can be easily concluded that we should choose the eigenvector associated to the smallest eigenvalue. The cost function can be written as follows

$$C = \theta^T \mathbf{M}^T \mathbf{M} \theta = \lambda^2 \theta^T \theta = \lambda^2 \tag{3.43}$$

The last quality comes from the fact that we are using normalized eigenvectors $\theta^T\theta = 1$.

The method we have described is very simple. We compute the eigenvalues and eigenvectors of $\mathbf{M}^T\mathbf{M}$ and choose the eigenvector associated to the smallest eigenvalue[6]. Figure 3.4 shows the alignment of an image obtained with a web camera with a drawing (regular grid). The alignment was done using a projective transformation and four marks (the grid corners). We have used the minimum number of marks required to estimate this transformation. Some matching errors can be noticed in the middle of the grid probably due to the deformation of the paper sheet or nonlinearity of the lenses.

Matching

The previous methods assume that the marks $\mathbf{x}, \mathbf{y}$ detected in both images are manually matched. How should we proceed when it is time consuming to match the marks?

There is no efficient algorithm to compute the optimal solution. This is still an open problem. To solve this problem in practice we resort to suboptimal methods such as the RANSAC.

[6]the eigendecomposition can be easily done in Matlab using the following command [l,v]=eig(M'M)

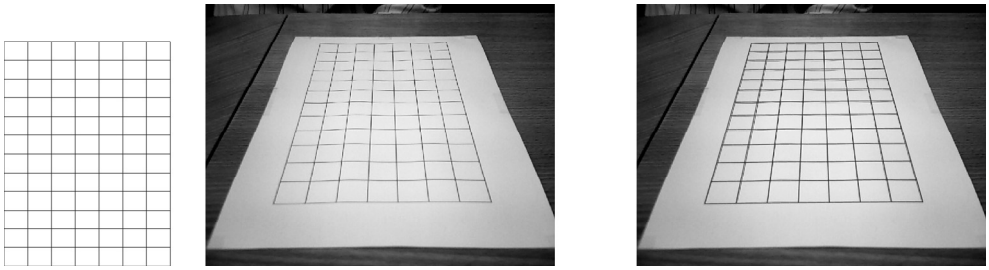


Figure 3.4: Image alignment: regular grid (left) observed image (center) and aligned images using a projective transformation (right)

The RANSAC method was proposed by Fishler and Bolles [18] performs many (hundreds) of tentative matches and chooses the best at the end.

It starts by randomly choosing the minimal set of points needed to estimate the transform and obtain an estimate for the unknown parameters. For example, in the case of the translation transform we would chose one point from each image.

In a second step we transform all the marks of the first image with the estimated transformation and check how many of them are close to the marks detected in the second image. This is done by computing the distance from the transformed points of the first image $T(\mathbf{x}_i)$ to the nearest points from the second image $\mathbf{y}_j$. The number of matches is called the support of the model.

This procedure is repeated many times. The model with highest support is chosen at the end.

Robustness

The least squares method used in the section is not robust in the presence of outliers i.e., wrong observations which are not explained by the model. The estimates produced by the least squares methods are severely affected by the outliers since the quadratic cost $\|\mathbf{y}_i - \mathbf{W}(\mathbf{x}_i, \mathbf{p})\|^2$ assigns a high penalty to these points and these points dominate the optimization procedure. A single outlier is enough to jeopardize the estimation procedure. Figure ? shows the least squares estimate of a line with one outlier.

To solve this problem we must use robust estimation methods. The RANSAC method is one possible solution for this problem since it is able to ignore a high percentage of outliers with a small degradation of quality. Another alternative would be the use of a different cost

function

$$C = \sum_i \rho(\mathbf{y}_i - \mathbf{W}(\mathbf{x}_i, \mathbf{p})) \tag{3.44}$$

where ρ is a penalty function growing slower that the quadratic penalty when the argument tends to infinity e.g., $\rho(x) = \tan^{-1}(\|x\|)$.

The criterion main difficulty concerns the minimization of this criterion which can only be one using numeric iterative algorithms. A good introduction to robust estimation methods in computer vision can be found in [1].

3.4 Alignment without marks*

This section is based in [2]. We wish to align two or more images without using marks. In this case geometric criteria are no longer valid since we do not know any marks. A different criterion is used instead based on a color constancy hypothesis. We assume that homologue points have similar intensity (color).

This can be done using a quadratic criterion

$$E(\theta) = \sum_{\mathbf{x}} (T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \theta))^2 \tag{3.45}$$

The geometric transformation between a pair of images $\mathbf{W}(\mathbf{x}; \theta)$ is estimated minimizing E.

The assumptions we are making are strong in some applications. Intensity (color) constancy is not true in practice if the camera or the object undergoes a large movement. Second the motion model is usually not valid for all the image points. In this case, the method can still by applied in small regions e.g., 5×5 blocks.

The minimization of E is a non linear problem. It is therefore necessary to adopt recursive numeric algorithms to obtain the motion model. In the sequel we describe the well known Lucas-Kanade method.

Lucas-Kanade method assumes that the current estimate of θ is slightly modified

$$\theta \leftarrow \theta + \mathbf{\Delta}\theta \tag{3.46}$$

If $\mathbf{\Delta}\theta$ is small we can approximate $I(\mathbf{W}(\mathbf{x}, \theta + \mathbf{\Delta}\theta)$ by the first terms of the Taylor series

$$I(\mathbf{W}(\mathbf{x}; \theta + \mathbf{\Delta}\theta) = I(\mathbf{W}(\mathbf{x}; \theta)) + \nabla I(\mathbf{W}(\mathbf{x}; \theta))^T \frac{\partial \mathbf{W}(\mathbf{x}; \theta)}{\partial \theta} \mathbf{\Delta}\theta \tag{3.47}$$

Replacing this in the energy expression we obtain

$$E = \sum_{\mathbf{x}} \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \theta)) - \nabla I(\mathbf{W})^T \frac{\partial \mathbf{W}}{\partial \theta} \mathbf{\Delta}\theta \right]^2 \tag{3.48}$$

Lucas-Kanade Algorithm

iterate until θ converges

- transform I through $\mathbf{W}(\mathbf{x};\theta)$ to obtain $I(\mathbf{W}(\mathbf{x},\theta))$
- compute the error image $T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\theta)$
- transform gradient ∇ through $\mathbf{W}(\mathbf{x};\theta)$ to obtain $\nabla I(\mathbf{W}(\mathbf{x};\theta))$
- compute the images $\nabla I(\mathbf{W}(\mathbf{x};\theta))^T \frac{\partial \mathbf{W}}{\partial \theta}$
- compute matrix $\mathbf{R}$
- compute vector $\mathbf{r}$
- compute $\mathbf{\Delta}\theta = \mathbf{R}^{-1} r$ and update $\theta \leftarrow \theta + \mathbf{\Delta}\theta$

Figure 3.5: Lucas-Kanade algorithm

The optimization of E with respect to $\mathbf{\Delta}\theta$ is easy now. Computing the derivative and making it equal to zero leads to

$$\sum_x \left(\nabla I(\mathbf{W})^T \frac{\partial \mathbf{W}}{\partial \theta} \right)^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\theta)) - \nabla I(\mathbf{W})^T \frac{\partial \mathbf{W}}{\partial \theta} \mathbf{\Delta}\theta \right] = 0 \tag{3.49}$$

This is a system of linear equations

$$\mathbf{R}\mathbf{\Delta}\theta = \mathbf{r} \tag{3.50}$$

where

$$\mathbf{R} = \sum_{\mathbf{x}} \frac{\partial \mathbf{W}}{\partial \theta}^T \nabla I(\mathbf{W}) \nabla I(\mathbf{W})^T \frac{\partial \mathbf{W}}{\partial \theta} \tag{3.51}$$

$$\mathbf{r} = \sum_{\mathbf{x}} \frac{\partial \mathbf{W}}{\partial \theta}^T \nabla I(\mathbf{W}) (T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\theta)) \tag{3.52}$$

Lukas Kanade algorithm is summarized in figure 3.5. Figure 3.6 shows several steps involved in the update of θ through Lucas-Kanade method. This figure was extracted from [2].

Some comments are required. We have assumed until now that $I, \nabla I$ are continuous images. The steps involving the warping of the image must be performed using interpolation algorithms with sub pixel accuracy. The recursion is computational demanding since the warping operations associated to the transformation $\mathbf{W}(\mathbf{x},\theta)$ must be performed at each iteration. There are alternative approaches which are computationally more efficient (see [2]).

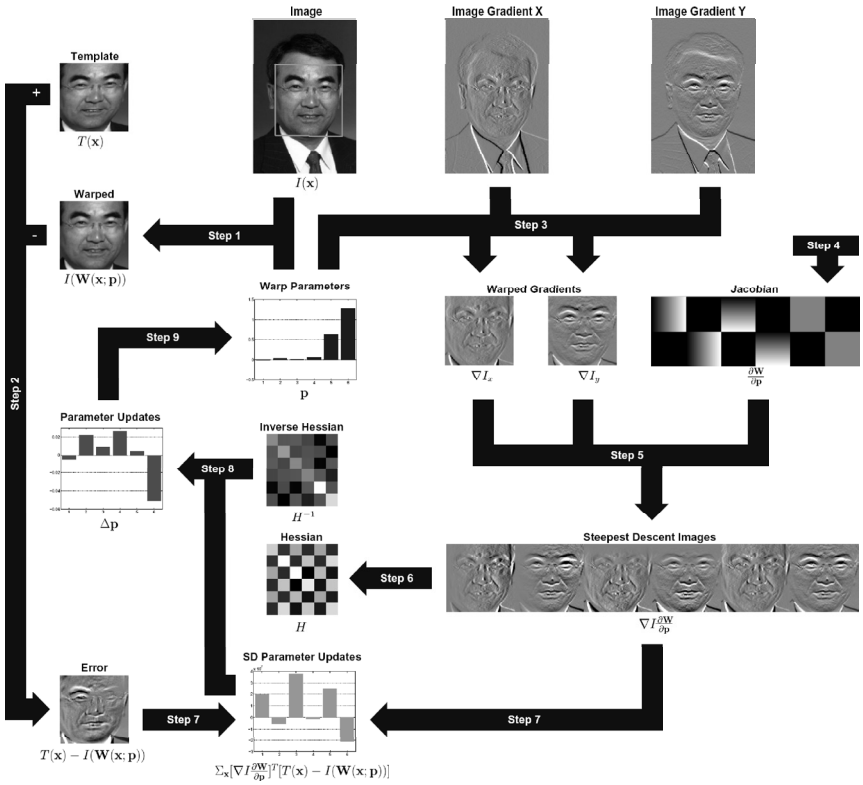


Figure 3.6: Lucas Kanade algorithm: operations involved in each iteration. Extracted from [2].

Let us see a simple example.

Example - translation

Suppose the motion model is a simple translation $\mathbf{W}(\mathbf{x}, \theta) = \mathbf{x} + \mathbf{t}, (\mathbf{p} = \mathbf{t})$. The derivative of $\mathbf{W}$ with respect to θ is the identity matrix.

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \mathbf{I} \tag{3.53}$$

In each iteration the translation is estimated solving a system of 2 equations $\mathbf{Rt} = \mathbf{r}$ where

$$\mathbf{R} = \sum_x \nabla I(\mathbf{W}) \nabla I(\mathbf{W})^T \qquad \mathbf{r} = \sum_x \nabla I(\mathbf{W})(T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p})) \tag{3.54}$$

3.5 Discussion

The alignment methods based on dense sets of points do not require point matching which is usually error prone. There are however other difficulties: a more difficult optimization and lack of robustness with respect to outliers and initialization.

The cost functional is a non convex function and has several local minima. The estimates obtained by numeric optimization algorithms get stuck in local minima. Therefore, the final estimate depends on the initialization. How can we solve this difficulty? the use of multiple scales has been an effective tool to deal with this problem [3]. A first estimate of the parameters is obtained using a coarse representation of the image. This estimate is refined using higher resolution images.

The estimation performed by the Lucas-Kanade method is sensitive to outliers. There are image regions where matching is not possible due to lack of information or because the model is no longer valid and must be replaced by another model. These regions significantly degrade the estimates since we are using a quadratic criterion. Robust estimation methods have been proposed (e.g., see [4]).

Bibliography

[1] P. Meer, D. Mintz, and A. Rosenfeld. Robust regression methods for computer vision: a review. Intl. J. Comp. Vis., 6(1):59–70, 1991.

[2] Simon Baker, Iain Matthews, Lucas-Kanade 20 Years On: A Unifying Framework: Part 1, CMU-RI-TR-02-16

[3] Pedro Aguiar, **COMPLETAR**

[4] M. J. Black, P. Anandan, The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields, Computer Vision and Image Understanding, CVIU, 63(1), pp. 75-104, Jan. 1996.

[5] I. Dryden, K. V. Mardia, Statistical Shape Analysis, Wiley, 1998

[6] D. Marr Vision, Freeman, 54-78, 1982.

[7] J. Canny A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, Nov. 1986.

[8] S. M. Smith, J. M. Brady, SUSAN A New Approach to Low Level Image Processing, International Journal of Computer Vision, Vol. 23, 45-78, 1997

[9] S. Konishi, A. L. Yuille, J. M. Coughlan, S. Chun Zhu, Statistical Edge Detection: Learning and Evaluating Edge Cues, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, NO. 1, JANUARY 2003.

[10] ttp://www.cns.nyu.edu/ eero/texture/ [**links sobre textura**]

[11] ihran Tuceryan, Anil Jain, Texture Analysis, The Handbook of Pattern Recognition and Computer Vision (2nd Edition), by C. H. Chen, L. F. Pau, P. S. P. Wang (eds.), pp. 207-248, World Scientific Publishing Co., 1998.

[12] D Cheng, XH Jiang, Y. Sun, Jingli Wang, Color image segmentation: advances and prospectus, Pattern Recognition, 34, 2259-2281, 2001.

[13] J. Canny A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, Nov. 1986.

[14] Ma, S. Soatto, J. Kosecká, S. S. Sastry, An Invitation to 3-D Vision. From Images to Geometric Models, Springer 2004.

[15] David G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, 60, 2, 91-110, 2004.

[16] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(5):530-534, 1997.

[17] D. Comaniciu, P. Meer: Mean Shift: A Robust Approach toward Feature Space Analysis, IEEE Trans. Pattern Analysis Machine Intelligence, Vol. 24, No. 5, 603-619, 2002.

[18] M. A. Fischler and R. C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". Comm. of the ACM 24: 381–395.

Chapter 4

Image Representation and Models

4.1 Multi-scale representations

An image contains objects of different sizes e.g., large buildings and smaller people. Small objects are best characterized if the camera sticks close to them while large objects should be observed at mutch larger distances. This means that we should characterize different objects at different scales, the scale being related to the distance from the camera to the object.

What is a multi-scale representation? a multi scale representations of an image I is a description of I which allows the analysis of the image at different scales, ranging from coarse scales to fine ones. An example of such a representation is the scale-space concept and the Gaussian pyramid considered in the next sections.

4.2 Scale-Space and Gaussian Pyramid

Let $I(x)$ be continuous image and let us define a distorted image $\mathcal{I}(\mathbf{x}; s)$ obtained by smoothing the image I with a Gaussian kernel.

$$\mathcal{I}(\mathbf{x}; s) = I(\mathbf{x}) * G_s(\mathbf{x}) \tag{4.1}$$

$$G_s(\mathbf{x}) = \frac{1}{2\pi s^2} e^{-\frac{\|x\|^2}{2s^2}} \tag{4.2}$$

Figure 4.1: Scale space images for $s = 0, 2, 4, 8, 16$

The convolution destroys the details smaller than s. The parameter s is therefore a scale parameter and the function $\mathcal{I}(\mathbf{x}; s)$ is called the **scale-space representation** of I.

Figure 4.1 shows some slices of $\mathcal{I}(\mathbf{x}; s)$ for different values of s showing the way small details are being lost.

When we work at coarse scales (large s) we need less information to represent the image. We can therefore devise ways of reducing the amount of information at coarse scales. This can be done by making a change of variable.

$$\mathcal{J}(\mathbf{x}; s) = \mathcal{I}(s\mathbf{x}; s) \tag{4.3}$$

If the size of $\mathcal{I}(\mathbf{x}; s)$ is $T \times T$ the size of $\mathcal{J}(\mathbf{x}; s)$ for a given s is $\frac{T}{s} \times \frac{T}{s}$. This is called a **pyramid**. The use of multiple scales improves the performance of the image analysis algorithms which can be tailored for a specific scale and reduces the computation effort since coarser scales can be processed much faster.

In practice we use discrete images and discrete scales values e.g., $s = 2^j$. The Gaussian pyramid can be defined by low pass filtering the image and down sampling by a factor of 2 to reduce the size of the image to one half at each iteration. Let I be a $N \times N$ discrete image. We assume that $N = 2^M$ for the sake of simplicity. We initialize the level 0 of the pyramid as

$$I_0(\mathbf{k}) = I(\mathbf{k}) \tag{4.4}$$

and recursively compute the other levels by smoothing the previous level and resampling the

Figure 4.2: Gaussian Pyramid, level $0, 1, 2, 3, 4$

smoothed image using with a $2:1$ factor

$$J_i(\mathbf{k}) = I_i(\mathbf{k}) * G_\sigma(\mathbf{k}) \qquad I_{i+1}(\mathbf{k}) = J_i(2\mathbf{k}) \qquad i = 1, \ldots, M \tag{4.5}$$

The sequence of images $I_0, I_1, \ldots, I_M$ of sizes $2^M \times 2^M, 2^{M-1} \times 2^{M-1}, \ldots, 1 \times 1$ define the **Gaussian pyramid** for the discrete image I. Figure 4.2 shows 4 levels of the Gaussian pyramid. The original image is shown on the left.

The Gaussian pyramid has one drawback. It increases the amount of memory needed to store the image. This leads to the following question: *can we build a multi-scale representation without increasing the number of coefficients?* This question will be answered using Haar functions and wavelet theory.

4.3 Haar Representation

Let us consider a 1D approximation problem first. Suppose we want to approximate a 1D function $f : \mathbb{R} \to \mathbb{R}$ by a piecewise constant function f_0 such that f_0 is constant between any two consecutive integers $[k, k+1[$. We can write this approximation as

$$f_0(x) = \sum_{k=-\infty}^{+\infty} c_k \phi(x) \tag{4.6}$$

where $\phi(x)$ is the box function (see Figure 4.3 left)

$$\phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

and c_k is the average value of f in the k-th interval $[k, k+1[$.

We can improve the accuracy of the piecewise approximation by considering smaller intervals e.g., by splitting each interval into 2^j subintervals. Every time we split the intervals we are

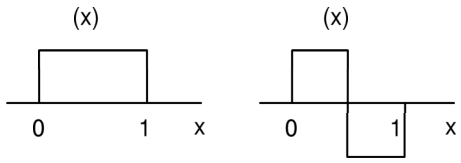


Figure 4.3: Haar basis functions: a) Haar scaling function; b) Haar wavelet

changing the scale. If we repeat this procedure we obtain a multi-scale approximation of f with an accuracy as high as desired. The index j is the scale level or resolution. The approximation obtained at level j is[1]

$$f_j(x) = \sum_{k=-\infty}^{+\infty} c_{jk}\phi(2^j x - k) \tag{4.8}$$

The basis functions are box functions compressed by a factor of 2^j i.e., with a shorter duration and shifted.

Figure 4.4 shows the multi-scale approximation of a speech signal by piecewise constant functions for $j = 3, 5, 7$ and figure 4.5 shows the evolution of the norm of the approximation error when the number of levels j increases

$$D(j) = \frac{||e_j||}{||f||} \qquad e_j = f - f_j \tag{4.9}$$

The error tends to zero as expected.

The representation (**??**) is very useful but it has a drawback. It is not recursive. Every time we want to increase the resolution level j we have to define a new set of basis functions and to compute all the coefficients again. **Can we avoid this?** The answer is yes and it is very simple in this case.

Let us consider the simplest case of a constant approximation $f_0(x)$ in the interval $[0, 1[$. If we want to split this interval in two we only have to consider a new basis function

$$\psi(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2} \\ -1 & \frac{1}{2} \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

and multiply it by an appropriate coefficient: $f_1(x) = f_0(x) + d\psi(x)$. This idea applies if we want to compute f_{j+1} from f_j in $\mathbb{R}$. We just have do add a new set of base function

$$C_j = \{\psi(2^j x - k)\} \tag{4.11}$$

[1] the functions are usually multiplied by a gain 2^j to keep their energy equal to 1. We have omitted the normalization factor for the sake of simplicity.

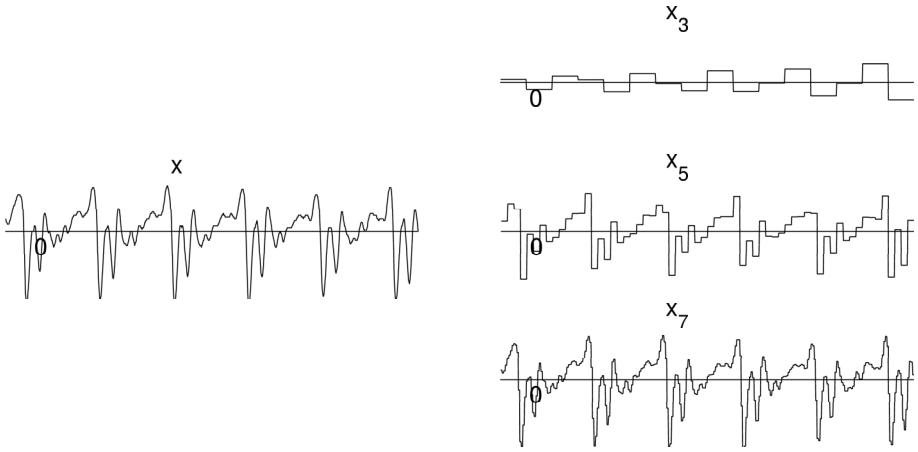


Figure 4.4: Hierarchical approximation of a speech signal using piecewise linear functions ($j = 3, 5, 7$)

Therefore,

$$f_{j+1}(x) = f_j(x) + d_j(x) \qquad d_j = \sum_{k=-\infty}^{+\infty} d_{jk}\psi(2^j x - k) \tag{4.12}$$

The signal f_j is called the approximation of level j and d_j is called the detail. The functions $\psi(2^i x - k)$ are called the $Haar functions$ and $\psi(x)$ is called the *Haar wavelet*. It can be shown that they are orthogonal and they span the $L^2(\mathbb{R})$ space, the space of functions with finite energy. Therefore the approximation function $f_j \to f$ when j tends to infinity.

Combining these equations we obtain

$$f_j(x) = f_0(x) + \sum_{i=0}^{j-1} \sum_{k=-\infty}^{+\infty} d_{ik}\psi(2^i x - k) \tag{4.13}$$

The function f_0 can also be expressed in terms of wavelets if the scale variable i starts at $-\infty$. We obtain two alternative ways to express f_j using scaled versions of ϕ and ψ

$$f_j(x) = \sum_{k=-\infty}^{+\infty} c_{jk}\phi(2^j x - k) \tag{4.14}$$

$$f_j(x) = c + \sum_{i=-\infty}^{j-1} \sum_{k=-\infty}^{+\infty} d_{ik}\psi(2^i x - k) \tag{4.15}$$

This decomposition can be interpreted in terms of subspaces. The function f_j is a linear combination of linearly independent functions $\phi(2^j x - k)$. This means that f_j belongs to a subspace V_j span by the base

$$B_j = \{\phi(2^j x - k), k \in \mathbb{Z}\} \tag{4.16}$$

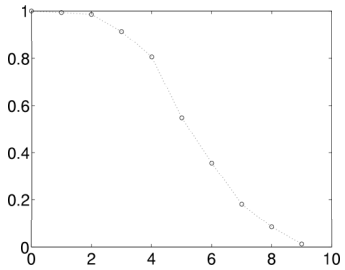


Figure 4.5: Norm of the approximation error as a function of the number of levels

ϕ(x+1)				ϕ(x)				ϕ(x-1)			
ψ(2x+2)		ψ(2x+1)		ψ(2x)		ψ(2x-1)		ψ(2x-2)		ψ(2x-3)	
ψ(4x+4)	ψ(4x+3)	ψ(4x+2)	ψ(4x+1)	ψ(4x)	ψ(4x-1)	ψ(4x-2)	ψ(4x-3)	ψ(4x-4)	ψ(4x-5)	ψ(4x-6)	ψ(4x-7)

Figure 4.6: Slots associated to hierarchical basis functions (3 levels)

The subspaces V_j are nested i.e.,

$$V_{-1} \subset V_0 \subset V_1 \subset V_2 \dots \tag{4.17}$$

The subspaces of lower resolution are contained in the subspaces of higher resolution. Since

$$f_{j+1} = f_j + d_j \tag{4.18}$$

where the detail d_j is a signal which belongs to the subspace $W_j = span\{\psi(2^j x - k)\}$. Each element of V_{j+1} is a sum of an element of V_j with an element of W_j. Therefore,

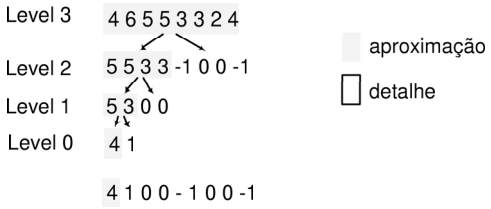
$$V_{j+1} = V_j \oplus W_j \tag{4.19}$$

where $\oplus$ is the direct sum of subspaces.

What is the relationship between the coefficients of both decompositions (4.14), (4.15)? The two sets of coefficients are closely related. If we know the coefficients c_j we can compute the coefficients of the approximation and detail of lower levels as follows

$$c_{jk} = \frac{1}{2}(c_{j+1,2k} + c_{j+1,2k+1}) \qquad d_{jk} = \frac{1}{2}(c_{j+1,2k} - c_{j+1,2k+1}) \tag{4.20}$$

For example given the coefficients $c_{3k} = (46553324)$ we can easily compute the approximation and detail coefficients for each level by computing sums and differences



$c_{00} = 4, d_{00} = 1, d_{1k} = (00), d_{2k} = (-100 - 1)$.

These ideas con be extended to a broader class of hierarchical representations as we will now discuss.

4.4 Wavelet Transform

4.4.1 Multiresolution Approximation

The Wavelet transform aims to approximate signals at different resolution levels, using scaled and shifted versions of a unique basis function called mother wavelet. Figure 4.4.1 shows an image (left) and its decomposition using 2 levels of the wavelet transform (right). The sub images represent the lower resolution approximation (up/left) and the six detail images i.e., information needed to reconstruct the original image. The process can be repeated. The low resolution image can decomposed again and replaced by four sub images. This allows to create a multi resolution representation which keeps the amount of coefficients constant. The wavelet theory is closely related to several topics in mathematics and signal processing: multi-resolution approximation, functional analysis and multi-rate filter banks. Excellent accounts on wavelets can be found in [1, 2, 3, 4].

Let us suppose we want to approximate a function $f(x) \in L^2(\mathbb{R})$ where $L^2(\mathbb{R})$ is the space of functions with finite energy. We will consider a set of nested subspaces of increasing resolution $\{V_i, i \in \mathbb{Z}\}$

$$\ldots V_{-1} \subset V_0 \subset V_1 \subset V_2 \ldots \tag{4.21}$$

such that if a function $f(x) \in V_j$ then $f(2x) \in V_{j+1}$. This means that if a function $f(x)$ belongs to V_j the function obtained by compressing it by a factor of 2 belongs to the next subspace.

Furthermore we will assume that the limiting set

$$\lim_{j \to \infty} V_j \tag{4.22}$$

Figure 4.7: Wavelet transform: approximation (upper left block) and 6 details

is dense in $L^2(\mathbb{R})$ i.e., any function $f \in L^2(\mathbb{R})$ can be approximated as well we wish by a function $f_j \in V_j$ provided that we choose j high enough.

Since the subspaces V_j are nested we can write each of them as a direct sum of subspaces

$$V_j = V_{j-1} \oplus W_{j-1} \tag{4.23}$$

where V_{j-1} is the lower resolution space and W_{j-1} is a subspace of detail. Therefore each function $f_j \in V_j$ can be decomposed in a unique way as

$$f_j = f_{j-1} + d_{j-1} \tag{4.24}$$

where $f_{j-1} \in V_{j-1}$ and $d_{j-1} \in W_{j-1}$. These functions are called the approximation and detail at resolution $j-1$.

How can we compute the approximation and the detail at several resolutions? The answer is easy if we know an orthogonal bases $\phi_{jk}(x), \psi_{jk}(x)$ for the subspaces V_j, W_j. Since the subspaces V_j at different scales are related by dilations by power of 2 it would be nice if the basis functions at different scales were also related by scaling and shifting operations. This problem is answered by the next theorem.

Theorem [Mallat]:

Given a multi resolution approximation $\{V_j, j \in \mathbb{Z}\}$ there exists two unique functions

$$\phi(x) \in L^2(\mathbb{R}) \qquad \psi(x) \in L^2(\mathbb{R}) \tag{4.25}$$

called **scaling function** and **mother wavelet** such that $\{2^{\frac{j}{2}}\phi(2^j x - k)\}$ is an orthonormal base for V_j and $\{2^{\frac{j}{2}}\psi(2^j x - k)\}$ is an orthonormal base for W_j.

For the sake of simplicity we will use the following notation

$$\phi_{jk}(x) = 2^{\frac{j}{2}}\phi(2^j x - k) \tag{4.26}$$

$$\psi_{jk}(x) = 2^{\frac{j}{2}}\psi(2^j x - k) \tag{4.27}$$

for the basis functions of V_j, W_j and the functions $\psi_{ij}(x)$ are called **wavelets**.

If we have orthonormal bases for the subspaces, the projections of a function f on V_j and W_j are given by

$$f_j(x) = \sum_k c_{j,k}\phi_{j,k}(x) \qquad c_{j,k} =< f, \phi_{j,k} > \tag{4.28}$$

$$d_j(x) = \sum_k d_{j,k}\psi_{j,k}(x) \qquad d_{j,k} =< f, \psi_{j,k} > \tag{4.29}$$

4.4.2 Fast Wavelet Transform

Equations (4.28,4.29) are not practical since they involve the computation of an integral for each coefficient at each scale. There are faster ways to compute the coefficients using the so-called fast wavelet transform.

We have not used yet the fact that the base functions at different scales are related by a contraction. This leads to important properties which are at the heart of the fast wavelet transform algorithm. The scaling function and the mother wavelet belong to V_0 and W_0, respectively. Therefore they are also members of the higher resolution subspace V_1 and can be expressed in terms of its basis functions

$$\begin{aligned} \phi(x) &= \textstyle\sum_k h_0(k)\phi_{1,k}(x) \qquad \text{(self-similarity)} \\ \psi(x) &= \textstyle\sum_k h_1(k)\phi_{1,k}(x) \end{aligned} \tag{4.30}$$

where

$$h_0(k) =< \phi(x), \phi_{1,k}(x) >= \sqrt{2}\int \phi(x)\phi(2x-k)dx \tag{4.31}$$

$$h_1(k) =< \psi(x), \phi_{1,k}(x) >= \sqrt{2}\int \psi(x)\phi(2x-k)dx \tag{4.32}$$

Using this result, it is easy to show that the approximation and detail coefficients at consecutive levels are related by (see Appendix)

$$c_{j-1,k} = \sum_n h_0(n-2k)c_{jn} \qquad d_{j-1,k} = \sum_n h_1(n-2k)c_{jn} \tag{4.33}$$

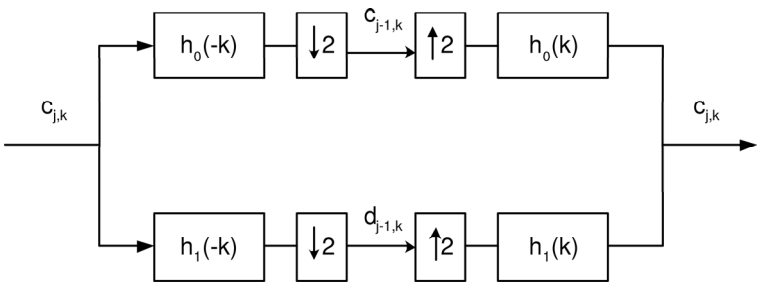


Figure 4.8: Fast wavelet transform: multi rate filter bank with perfect reconstruction

$$c_{jn} = \sum_k h_0(n-2k)c_{j-1,k} + \sum_k h_1(n-2k)d_{j-1,k} \tag{4.34}$$

The first equation computes the pyramid from the high resolution coefficients i.e., it computes successive coarse approximations and details of the signal from the high resolution coefficients. The second equation does the opposite. It computes the high resolution approximation from the low resolution coefficients and details.

We can easily express these equations in terms of convolution, up sampling and down sampling operations defined as follows

$$[\uparrow]c(n) = \begin{cases} c(n/2) & n \text{ even} \\ 0 & n \text{ odd} \end{cases} \qquad [\downarrow]c(n) = c(2n) \tag{4.35}$$

The previous equations can be written as

$$c_{j-1,k} = [\downarrow](h_0(-k) * c_{jk}) \qquad d_{j-1,k} = [\downarrow](h_1(-k) * c_{jk}) \tag{4.36}$$

$$c_{jk} = h_0(k) * [\uparrow]c_{j-1,k} + h_1(n) * [\uparrow]d_{j-1,k} \tag{4.37}$$

These equations define a multi rate filter bank which implements both steps (see Figure 4.8). This filter bank comprises two parts: the analysis and the synthesis which correspond to the computation of the lower resolution coefficients from high resolution coefficients and the opposite. The wavelet transform maps a set of high resolution coefficients into a set of coefficients corresponding to different resolution level.

This established a close link between wavelet theory and multi rate filter bank theory. The wavelet decomposition is associated with a perfect reconstruction multi rate filter bank.

4.4.3 Representation of discrete images

How do we apply the wavelet decomposition to discrete signals?

The wavelet decomposition provides a hierarchical representation of continuous signals. However we can use it to represent discrete signals as well. This can be done by initializing the coefficients of the j level with samples of the discrete signal, $c_{j,n} = f(n)$, and then apply the fast wavelet transform to obtain the approximations and details at lower resolution levels.

How can we use this transform with images? This question can be solved in an easy way if we use separable basis functions. This amounts to computing the wavelet transform to each column of the image and the apply it again to each row of the transformed image.

4.5 Probabilistic Models

Probabilistic models are used in many image processing operations. Suppose we wish to separate objects of interest from a background image. To solve this problem we need a statistical model of the background image. in order to detect regions which do not fit the background model. This techniques is used in background subtraction methods. Other examples are provided by image denoising and restoration problems in which we want to improve the quality of images corrupted by noise. Suppose that the observed image is corrupted by additive noise

$$J = I + W \tag{4.38}$$

in which I is the original image, J the observed image and W is the noise. We need probabilistic models for the original image I and noise W in order to separate them. Without models we can not separate I from W since we only know their sum.

The basic question is the following: *given a pair of images we want to know which one has higher probability, in a certain context.* Figure 4.9 shows two binary images which represent moving objects in a scene. Which one is more probable? The answer is very easy in this case. The left image is the most probable. We can recognize the objects from their silluette. Prior information about the objects shape from several viewpoints is an important cue. A simpler criterion is that we have chosen the smoother image i.e., the image with less transitions. Can we include this kind of evaluation in a probability measure?

To characterize a random image $I = (I_1, \ldots, I_n)$ we need to know its joint probability distribution

$$p(I) = p(I_1, \ldots, I_n) \tag{4.39}$$

which depends on all its (thousands) variables. It is not possible to estimate this distribution from experimental data except if we make very strong hypothesis. This can be done if we are

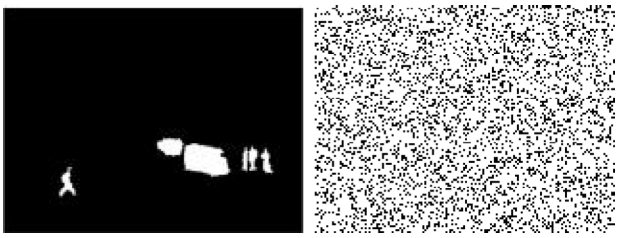


Figure 4.9: Are all the images equally probable?

working with a specific type of images (e.g., normalized faces). However, in the general case it is not possible to obtain accurate models in this way. Therefore, the probabilistic models used in image processing are poor models in the sense that they only try to represent a small subset of the properties of real images. Still, they play an important role in many applications.

4.6 First and second order histograms

The histogram is a simple way to characterize an image. To build a histogram we split the intensity range into a number of bins and count the number of pixels which fit in each bin. In this way we are estimating the probability distribution of isolated pixels $p(I_i)$ assuming that this distribution is valid for the whole image. Figure 4.10 shows the histograms associated to two images.

The histogram is a very crude representation. It neglects all spatial information and it also assumes stationarity. If the pixels were independent variables, the joint distribution could be computed multiplying these factors

$$p(I) = \prod_{i=1}^{k} p_i(I_i) \tag{4.40}$$

However this is not true. Images are highly correlated. A second step consists of studying pairs of pixels with a given spatial relationship e.g., neighboring pixels along horizontal lines. We can now define a scatter plot ploting all pairs of image intensities I_i, I_j such that i, j obey the geometric restriction. Figure 4.11 shows the scatter plots for the two previous examples.

We can characterize pairs of pixels using a 2D histogram which estimates the joint distribution of neighboring pixels $p(I_i, I_j)$. The relative position of pixels i, j is defined by the user e.g., pixels separated by a distance d along a given direction. The histogram is computed using all pairs of pixels with the same relative position belonging to the image or to a specific region of the image. We are therefore assuming that the image is a stationary process i.e., its statistical

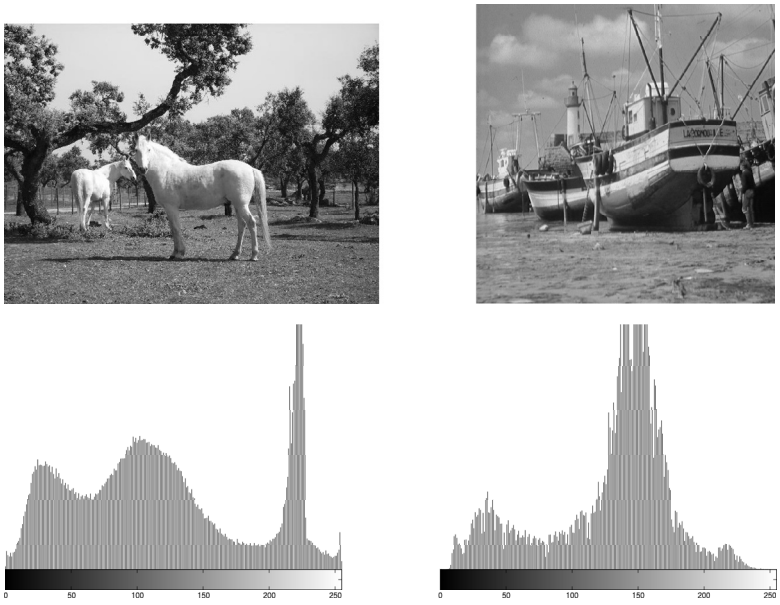


Figure 4.10: images (up) and intensity histograms (down)

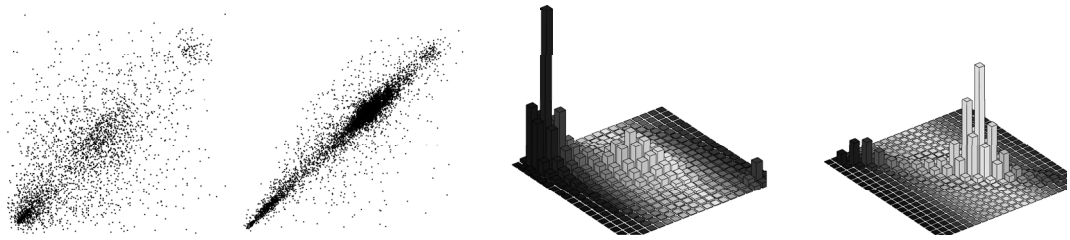


Figure 4.11: Scatter plots and 2d histograms

properties remain the same everywhere. Of course this is not always true. It is true if we are dealing with an aerial image of a forest but not true if we apply the method to the image of a face for example.

The 2D histogram of pairs of pixels is a way to characterize textures in a given region and has been used for texture recognition. The 2D histogram is often known as co-occurrence matrix in the context of texture analysis.

Unfortunately the application of 2D histograms is very restrictive and the idea can not be extended to characterize the whole image involving thousands of variables. *How should we proceed then?*

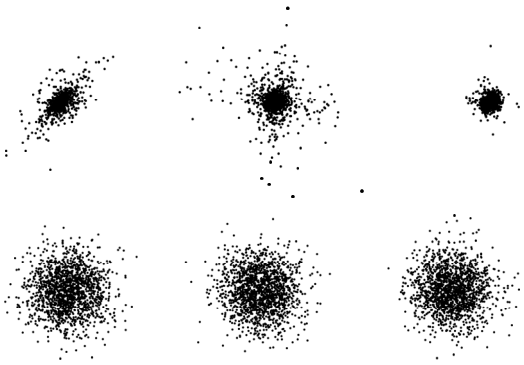


Figure 4.12: Scatter plots of detail coefficients for the boat (up) and noise (down) images: horizontal detail (left), vertical detail (center) diagonal detail (right). The plots were built using pairs of consecutive coefficients along the horizontal direction.

4.7 Wavelet models

Neighboring pixels usually have a strong dependence. Modeling this dependence is a difficult step in the design of a probabilistic model for the image. However, it has been found that the wavelet transform decorrelates the image since the detail coefficients are less correlated.

Figure 4.12 shows the scatter plot for the three details of the first level decomposition of images boat and horse?. We have considered pairs of neighbor coefficients along the horizontal direction for the three types of detail images. We can observe a strong concentration of points in the vicinity of the origin (many coefficients have a small amplitude). There is no apparent dependence between the coefficients and it is reasonable to assume that the coefficients of each detail image are approximately independent. The same can be said as a first approximation if we compare the coefficients of different detail images within the same scale or at different scales. If we stack all the detail coefficients in a single vector $d = (d_1, \ldots, d_m)$ we can assume that

$$p(d) = \prod_{i=1}^{n} p(d_i) \tag{4.41}$$

Since there is a 1-1 linear map between the detail coefficients and the image I

$$d = WI \tag{4.42}$$

with $\det(W) = 1$ then

$$p(I) = p(d) \tag{4.43}$$

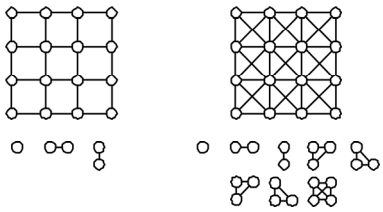


Figure 4.13: Image graphs: 4 and 8 neighborhood systems and corresponding cliques

4.8 Markov Random Fields

An image is a set of random variables organized in rows and columns (see Fig ?). There is dependence among these variables: if we know the value of some pixels we can predict the value of others. In fact there is a short range dependence and long range dependence mechanisms.

It is perhaps reasonable to assume that the influence of the whole image on a given pixel I_i can be discarded if we know its neighbors. This can be written as

$$p(I_i/I - \{I_i\}) = p(I_i/N_i) \tag{4.44}$$

where $I - \{I_i\}$ is the set of all image pixels except i and N_i are the neighboring pixels of pixel i. This is called the **Markov property**.

Let $\mathcal{G} = (S, N)$ be a graph consisting of a set of sites (nodes) S and a set of links $N = \{(i,j), i,j \in S\}$ The sites represent the pixel location and the links represent neighborhood connections. Figure 4.11 shows two graphs corresponding to two neighborhood systems: 4-neighborhood and 8-neighborhood. A group of nodes fully connected i.e., such that any pair of nodes is connected is called a *clique*. Figure 4.11 also shows the different types of cliques associated to the graph.

Let $I = (I_i, i \in S)$ be a set of random variables associated to the nodes of the graph. In our case I is an image and the nodes are the image pixels.

Definition [Markov Random Field] I is a Markov Random Field (MRF) iif (if and only if)

i) $P(I) > 0$, $\forall I$

ii) $p(I_i/I - \{I_i\}) = p(I_i/N_i)$, $\forall i \in S$ (Markov property)

The first is a technical assumption. The second is the Markov property which means that the vicinity of a pixel isolates the pixel from the rest of the image. To predict the value of a pixel we just have to know its neighbors.

This seems to be a reasonable assumption which takes pixel dependence into account as we wished but builds long range dependences using local dependences.

4.8.1 Gibbs Random Fields

Let us consider a related concept. A Gibbs random field (GRF) is a set of random variables $I = (I_i, i \in S)$ associated to a graph $\mathcal{G} = (S, N)$ such that $p(I)$ is a Gibbs distribution

$$p(I) = \frac{1}{Z} e^{-E(I)} \tag{4.45}$$

where $E(I)$ is an energy function defined as follows

$$E = \sum_{c \in C} V_c(I_c) \tag{4.46}$$

where C is the set of cliques of the graph, V_c is a function of the clique variables I_c called potential function and Z is a normalization constant called the partition function. This expression states that the energy is a sum of potential functions $V_c(I_c)$ associated to the cliques of the graph.

The clique potentials V_c are specified by the user or estimated from the data. Estimation is usually a difficult step in this type of models. The partition function is given by

$$Z = \sum_{I} e^{-E(I)} \tag{4.47}$$

It involves the sum of the exponential function for all possible images. There is no analytical expression for Z in most problems and it is impossible to evaluate it in a computer due to the huge number of terms involved in the sum. In most of the cases we do not know Z.

The probability distribution can be factorized as a product of terms each of them associated to a clique of the graph

$$p(I) = \prod_{c \in C} \Phi_c(I_c) \tag{4.48}$$

where

$$\Phi_c(I_c) \propto e^{-V_c(I_c)} \tag{4.49}$$

Example - Ising model

The Ising model was proposed in the context of statistical physics for the study of ferromagnetism[2]. It assumes that the image I is a binary random field with a Gibbs distribution

[2]Ernst Ising (1998-2000) was a German physicist who studied a model for ferromagnetism in is Ph.D. thesis. Ising was persecuted by the nazis during World War II and moved to the US after the end of the war. He taught physics and mathematics at Bladley university until his retirement.

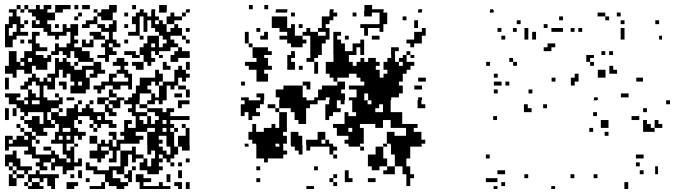


Figure 4.14: Realizations of the Ising MRF ($\beta = 0.8$)

with energy

$$E(I) = \beta \sum_{(i,j) \in N} \delta(I_i - I_j) \tag{4.50}$$

where δ is the impulse and N is the set of all 4-neighbor pixels. The clique potentials are defined as

$$V(i,j) = \beta \delta(I_i - I_j) \tag{4.51}$$

for all neighboring pairs of pixels. In the Ising model, the energy counts the number of horizontal and vertical transitions. A constant image has energy $E = 0$. The parameter β is defined as the inverse of the temperature and controls the average number of transitions in an image.

Let us consider an example

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{matrix}$$

The energy is $E = 10\beta$.

Figure 4.14 shows images produced by this model for three values of β (pixels marked with 1 are displayed in black). The size the regions depends on β. This model is very simple but it is useful in several applications e.g., to represent moving regions on a background image or two types of land cover in aerial images.

4.8.2 Equivalence of MRF and GRF

Hammersley-Clifford proved that Markov random fields and Gibbs fields are equivalent i.e., a random image is a MRF on a graph $\mathcal{G}$ iif it has a Gibbs distribution

$$p(I) = \frac{1}{Z} e^{-E(I)} \tag{4.52}$$

where

$$E = \sum_{c \in C} V_c(I_c) \tag{4.53}$$

and C is the set of all cliques. This is called the Hammersley-Clifford theorem.

4.8.3 Simulation

It is often useful to simulate the MRF and generate realizations of the field variables using the Gibbs distribution. Two methods which are often used to simulate MRFs are the metropolis algorithm and the Gibbs sampler.

The metropolis algorithm produces a sequence of images as follows. Given an image I we introduce a random change

$$I \to I' \tag{4.54}$$

and compute the energy difference

$$\Delta E = E(I') - E(I) \tag{4.55}$$

If ΔE is negative the new image I' is accepted, otherwise I' is randomly accepted with probability $P = e^{\Delta E}$ between 0 and 1. If P is close to 1 we accept I' with a high probability. If P is close to zero we reject I' most of the time and keep the previous image I. The random change can be done in many ways. We can make a local change (e.g., modify a single pixel) or a more global one (modify a group of pixels). The best strategy depends on the problem. However, it can be shown that under broad conditions the sequence of variables I, generated in this way, has an asymptotic Gibbs distribution (4.52).

To generate a realization of the MRF with distribution (4.52) we should perform a large number of iterations of the Metropolis algorithm and pick the last image. If we want to obtain several realizations of the random field we should not pick the images at consecutive iterations since they will be dependent. To obtain independent realizations we must perform several iterations between two consecutive realizations.

Metropolis Algorithm	**Gibbs Sampler**
initialization: initialize I randomly	initialization: initialize I randomly
iterate N times	iterate N times
random change: $I \rightarrow I'$	iterate for all pixels
energy variation: $\Delta E = E(I') - E(I)$	randomly change pixel i: $p(I_i/\bar{I}_i)$
decision: accept I' with prob. $P = min(1, e^{\Delta E})$	end
end	end
MRF realization: I	MRF realization: I

Table 4.1: Metropolis algorithm (left) and Gibbs sampler (right)

An alternative approach is the Gibbs sampler. The Gibbs sampler starts from an random initialization of the image and modifies one pixel at a time scanning the image by rows or columns. Each pixel is randomly modified keeping all the other constant. This is done by sampling the conditional distribution $p(I_i/\bar{I}_i)$ where $\bar{I}_i = I - \{I_i\}$. The conditional distribution is also a Gibbs distribution

$$p(I_i/\bar{I}_i) = \frac{1}{Z'} e^{\sum_{c \in C_i} V_c(I)} \tag{4.56}$$

where C_i is the set of cliques which contain the pixel i. The only variable in this expression which is allowed to vary is I_i since all the other are considered as constant and remain invariant. The partition function Z' can now be easily computed

$$Z' = \sum_{I_i} e^{\sum_{c \in C_i} V_c(I)} \tag{4.57}$$

the sum involves a single variable.

The Gibbs sampler is summarised in Table **??**. The convergence of the Gibbs sampler to the asymptotic distribution is usually faster that the convergence of the Metropolis algorithm but both methods are used in practice.

4.8.4 Optimization

Many problems in computer vision and image processing can be modeled by Markov random fields with Gibbs distribution

$$p(I|J) = \frac{1}{Z} e^{-E(I;J)} \tag{4.58}$$

where J is the available information (observations). We often want to know what is the most probable image I. This can be done by minimizing the energy

$$\hat{I} = \arg\min_{I} E(I; J) \tag{4.59}$$

In many cases a 4-neighborhood system is adopted and the energy function has two types of terms: terms associated to isolated pixels and pairs of pixels

$$E(I) = \sum_{i \in N} f_i(I_i) + \sum_{(i,j) \in N} f_{ij}(I_i, I_j) \tag{4.60}$$

The minimization of this energy is still a very difficult problem since the energy depends on a huge number of variables (typically hundreds of thousands) and it is a non convex function. Most optimization algorithms perform poorly in this type of problems and usually get stuck in local minima.

The optimization step has been a major bottleneck which prevents a widespread use of MRF in computer vision. Several techniques have been proposed to deal with this problem. We will briefly mention a few of them:

- Iterated Conditional Model (ICM)

  ICM is a very simple method. It basically optimizes one variable at a time keeping all the other constant. For example, if I is a binary image we visit each pixel at a time and choose the label (0 or 1) which minimizes the energy, freezing all the others. After visiting all the pixels we repeat the whole procedure starting from the image obtained in the previous iteration. The ICM method is simple but it usually gets stuck in local minima and may provide poor results.

- **Stochastic methods**

  The idea is also simple. Let us define an auxiliary MRF with Gibbs distribution

  $$p_{aux}(I) = \frac{1}{Z} e^{-\beta E(I;J)} \tag{4.61}$$

  If we increase β the distribution tends to zero except in the vicinity of the configurations which correspond to the global maximum (see Figure).

Evolution of the Gibbs distribution when β increases.

To obtain the global maximum the MRF is simulated using the Metropolis algorithm or the Gibbs sampler. The parameter β is slowly increased during the simulation. It can be shown that if we increase β slow enough the sequence of MRF generated in this way converges to the global maximum. These methods perform well but they are very slow.

- **Optimization on graphs**

  It has been recently proposed a class of optimization algorithms based on graphs. These methods perform better than the previous ones but their theory is still a research issue and it is outside the scope of this text. The interested reader may find an introduction to this topic in [**?**].

4.8.5 Parameter Estimation

The energy function often depends on parameters chosen by the user. For example, the Ising model depends on a parameter β which has to be estimated. More complex models depend on larger sets of parameters which we will denote by θ.

The estimation of the energy parameters from an image is a difficult problem since it is not easy to apply standard estimation methods such as the maximum likelihood method. The likelihood function is given by

$$p(I|\theta) = -\log Z(\theta) - \sum_{c \in C} V_c(I_c; \theta) \tag{4.62}$$

Unfortunately we do not have a closed form expression for the partition function nor an efficient way to compute it. The definition

$$Z(\theta) = \sum_{I} e^{-\sum_{c \in C} V_c(I_c;\theta)} \tag{4.63}$$

cannot be applied in practice since it involves the sum of the non-normalized distribution for all possible image configurations.

4.9 Appendix - Fast Wavelet Transform

Inner products of basis function

Inner products of basis functions at consecutive levels $< \phi_{j,p}, \phi_{j-1,q} >$, $< \phi_{j,o}, \psi_{j-1,q} >$ can be easily computed using the impulse responses h_0, h_1:

$$< \phi_{j,p}, \phi_{j-1,q} >= \int 2^{j-\frac{1}{2}}\ \phi(2^j x - p)\ \phi(2^{j-1}x - q)dx = \sqrt{2}\ \int \phi(2y + 2q - p)\ \phi(y)dy \quad (4.64)$$

$$< \phi_{j,p}, \psi_{j-1,q} >= \int 2^{j-\frac{1}{2}}\ \phi(2^j x - p)\ \psi(2^{j-1}x - q)dx = \sqrt{2}\ \int \phi(2y + 2q - p)\ \psi(y)dy \quad (4.65)$$

where $y = 2^{j-1}x - q$. Therefore,

$$< \phi_{j,p}, \phi_{j-1,q} >= h_0(p - 2q) \quad (4.66)$$

$$< \phi_{j,p}, \psi_{j-1,q} >= h_1(p - 2q) \quad (4.67)$$

Fast wavelet transform: analysis

The analysis step of the fast wavelet transform expresses the high resolution representation into a lower resolution representation. First, let us try to represent the basis functions of V_{j-1}, W_{j-1} as a linear combination of the basis functions of V_j

$$\phi_{j-1,k}(x) = \sum_n c_n \phi_{j,n}(x) \qquad \psi_{j-1,k}(x) = \sum_n d_n \phi_{j,n}(x) \quad (4.68)$$

where $c_n =< \phi_{j-1,k}, \phi_{j,n} >= h_0(n - 2k)$, $d_n =< \psi_{j-1,k}, \phi_{j,n} >= h_1(n - 2k)$ (see (4.66,4.67)).

The approximation and detail coefficients can be computed as follows

$$c_{j-1,k} =< f, \phi_{j-1,k} > \qquad d_{j-1,k} =< f, \psi_{j-1,k} > \quad (4.69)$$

Using (4.68) we obtain

$$c_{j-1,k} =< f, \phi_{j-1,k} >=< f, \sum_n h_0(n - 2k)\phi_{j,n} > \quad (4.70)$$

$$c_{j-1,k} = \sum_n h_0(n - 2k)c_{j,n} \quad (4.71)$$

and

$$d_{j-1,k} =< f, \psi_{j-1,k} >=< f, \sum_n h_1(n - 2k)\phi_{j,n} > \quad (4.72)$$

$$d_{j-1,k} = \sum_n h_1(n - 2k)c_{j,n} \quad (4.73)$$

as wished.

Fast wavelet transform: synthesis

Now we want to express ϕ_{jn} in terms of lower resolution basis functions

$$\phi_{jn}(x) = \sum_k c_k \phi_{j-1,k}(x) + \sum_k d_k \psi_{j-1,k}(x) \tag{4.74}$$

where $c_k =< \phi_{jn}, \phi_{j-1,k} >= h_0(n-2k), d_k =< \phi_{jn}, \psi_{j-1,k} >= h_0(n-2k)$ Since, $c_{j,n} =< f, \phi_{j,n} >$ we obtain

$$c_{j,n} =< f, \sum_k h_0(n-2k)\phi_{j-1,k} + < f, \sum_k h_1(n-2k)\psi_{j-1,k} > \tag{4.75}$$

$$c_{j,n} = \sum_k h_0(n-2k) < f, \phi_{j-1,k} > + \sum_k h_1(n-2k) < f, \psi_{j-1,k} > \tag{4.76}$$

$$c_{j,n} = \sum_k h_0(n-2k) c_{j-1,k} + \sum_k h_1(n-2k) d_{j-1,k} \tag{4.77}$$

as we wished.

Bibliography

[1] S. Mallat, A Wavelet Tour of Signal Processing, Academic Press, 2nd edition, 1999.

[2] S. Mallat, A theory for multi resolution signal decomposition : the wavelet representation, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 11, p. 674-693, July 1989.

[3] R. A. Gopinath, C. S. Burrus, Wavelets and Filter Banks, in Wavelets: a Tutorial in Theory and Applications, C. K. Chui (ed.), Academic Press.

[4] T.Li, Q. Li, S. Zhu, M. Ogihara, A Survey on Wavelet Applications in Data Mining, 2002

Chapter 5

Learning

5.1 Introduction

Learning methods play a central role in image and processing and vision and can be applied to a variety of problems ranging from texture analysis to image segmentation, object detection and tracking.

Let us consider a couple of examples. Suppose we observe the motion of an object in an image. How can we predict the object position in the next frame and learn to do that in an automatic way? Suppose we observe an aerial image with different types of textures. How can we learn the main features of each region and find its boundaries?

Learning can be done using deterministic or statistical methods. In both cases we want to predict a variable y given an observation x and try to do it by learning a function

$$\hat{y} = f(x, \theta) \tag{5.1}$$

which depends on an unknown parameter θ.

We will address both approaches through a set of simple problems.

5.2 Motion Prediction

Suppose we observe the motion of a point object in an image (e.g., a circular motion) and want to predict the position of the object in the next frame.

Let $\mathbf{x}(t)$ be the position of the point at frame t (t is an integer). We want to predict $\mathbf{y}(t) = \mathbf{x}(t+1)$. How can we learn this motion?

One strategy is the following. First we choose a model which we believe is rich enough to describe the motion

$$\hat{y} = f(x, \theta) \tag{5.2}$$

where θ is a set of unknown variables. In the case of point motion we could assume for example that the predicted coordinates $y_1(t), y_2(t)$ are related to the current coordinates $x_1(t), x_2(t)$ by

$$\begin{aligned} y_1(t) &= a_1 x_1(t) + a_2 x_2(t) + a_3 \\ y_2(t) &= a_4 x_1(t) + a_5 x_2(t) + a_6 \end{aligned} \tag{5.3}$$

with $\theta = (a_1 \dots a_6)$. Second, we need to learn the model parameters θ from the available data. This can be done by collecting a set of input-output pairs $\{(x(t), y(t)), t = 1, \dots, N\}$ and then adjust the unknown parameters θ by fitting the model to the data using a sum of squared errors criterion

$$E(\theta) = \sum_{i=1}^{N} \|y(t) - f(x(t), \theta)\|^2 \tag{5.4}$$

The parameters are estimated by minimizing this criterion

$$\hat{\theta} = \arg\max_{\theta} E(\theta) \tag{5.5}$$

This is called off-line training since we first collect the data and fit the model later. We could also do both things simultaneously i.e., update the model every time we receive new information. In this case we would be doing do an on-line training.

The third step consists of checking if the model is good enough. If it is not, we should try to improve it changing the model structure (5.2), the fitness criterion (5.4) or the optimization algorithm.

This method is called the *least squares method*[1] and it is fully deterministic.

Let us solve the prediction problem using the model 5.3. The model can be written as

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} x_1(t) & x_2(t) & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1(t) & x_2(t) & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_6 \end{bmatrix} \tag{5.6}$$

[1] the least squares method was proposed by Gauss in the 19th century, as a tool to predict the position of planets from experimental data

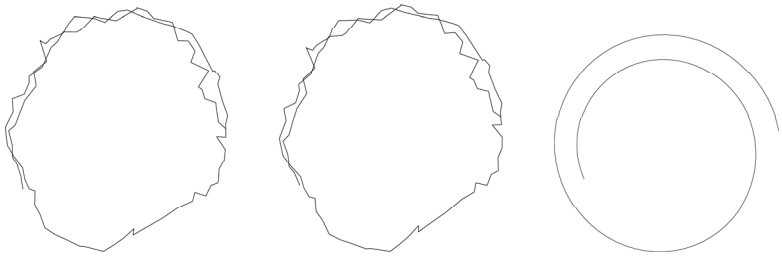


Figure 5.1: Motion prediction: original data (left), one step prediction (center) and prediction from the initial condition by recursive application of the one step predictor

Writing the equations for the observed instants of time we obtain

$$\begin{bmatrix} y_1(1) \\ y_2(1) \\ \vdots \\ y_1(N) \\ y_2(N) \end{bmatrix} = \begin{bmatrix} x_1(1) & x_2(1) & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1(1) & x_2(1) & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(N) & x_2(N) & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1(N) & x_2(N) & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_6 \end{bmatrix} \tag{5.7}$$

This can be written in a compact way as

$$M\theta = b \tag{5.8}$$

The quadratic cost functional is given by

$$E = (M\theta - b)^T (M\theta - b) \tag{5.9}$$

and can be easily minimized by computing the derivative with respect to θ and making it equal to zero. We obtain

$$M^T M\theta = M^T b \tag{5.10}$$

This is a system of 6 equations which can be easily solved.

Figure 5.1(left) shows 100 points of a circular trajectory with random disturbances. The motion model was then learned using this trajectory and the prediction estimates are shown at the center. To see how good is the motion, we generated the trajectory on the right recursively applying the predictor to predicted data.

$$\begin{aligned} y_1(t) &= a_1 t_1(t-1) + a_2 y_2(t-1) + a_3 \\ y_2(t) &= a_4 y_1(t-1) + a_5 y_2(t-1) + a_6 \end{aligned} \tag{5.11}$$

This trajectory was generated without data.

Least Squares Method

Goal: predict a variable y from x

Training data: $\{(x(t), y(t)), t = 1, \ldots, N\}$

1. Choose the model: $y = f(x, \theta)$, (θ unknown)
2. Choose the cost function: $E(\theta) = \sum_{i=1}^{N} \|y(t) - f(x(t), \theta)\|^2$
3. Estimate θ: $\hat{\theta} = \arg\max_\theta E(\theta)$
4. Validation: apply the model to new data and check if the desired accuracy was obtained.

Table 5.1: Least squares method

The least squares method is summarized in table 5.1.

The solution is very simple because the cost function is quadratic and the model is linear with respect to the unknown parameters. We can use more flexible models with nonlinear terms in x (e.g., polynomials)

$$\hat{y} = \sum_{i=1}^{p} a_i f_i(x; \theta) \tag{5.12}$$

and still obtain a linear set of equations for θ. Things become more complicated when the model is nonlinear in θ or the cost function is non quadratic. In these cases the optimization is often done using numeric recursive algorithms.

5.3 Texture classification

Let us consider a different problem. Suppose we have an image with two textures and want to automatically label each of them. We will make some simplifying hypothesis. We assume that the image I is a collection of $N \times N$ blocks each of them being filled by one of the textures.

To solve this problem we first have to characterize each of the textures i.e., we have to learn their properties. For the sake of simplicity we assume that there are training images of the two textures and we can use use these images to learn the texture properties. Second we wish to estimate the label $y \in \{0, 1\}$ associated to each block of the test image.

Figure 5.2 (left,center) shows the training images used to estimate the model. These textures were synthesized using the difference equation

$$I(m, n) = a_{10} I(m-1, n) + a_{01} I(m, n-1) + a_{11} I(m-1, n-1) \tag{5.13}$$

with the following parameters: $a_{01} = 0.4, a_{10} = .1, a_{11} = .5$ (texture 0), $a_{01} = 0.1, a_{10} =$

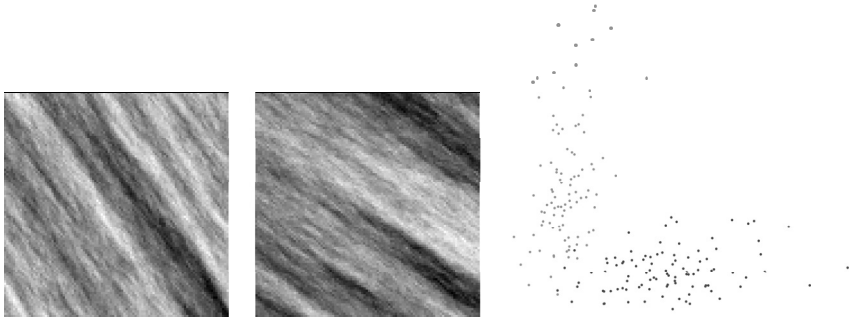


Figure 5.2: Texture learning: two textures (left, center) and scatter diagram of texture features computed in 10×10 blocks

$.4, a_{11} = .5$ (texture 1); $w(m,n)$ is white (uncorrelated) Gaussian noise with distribution $N(0,1)$. Figure 5.2 shows two images produced by this model. Both textures have zero mean and the same variance. These parameters are not able to discriminate the textures

Both textures have directional structure. Therefore, we will adopt two directional features based on intensity differences along different directions

$$x_1 = E\{\delta^2_{2,1}(m,n)\} \qquad x_2 = E\{\delta^2_{1,2}(m,n)\} \tag{5.14}$$

where δ_{ij} are intensity differences defined as follows: $\delta_{ij}(m,n) = I(m,n) - I(m-i, n-j)$. We compute a pair of features for each 10×10 block. The expected value is obtained averaging the squared differences in each block. Figure 5.2 (right) shows 100 feature points for each texture computed in 10×10 blocks. This scatter plot shows that we can discriminate both textures with few errors since the overlap of the two clouds is small.

The next question is: how do we classify texture blocks using this information? If we knew the probability distribution of the features associated to each class the answer would be simple. The optimal decision is based on the Bayes law

$$P(y|x) = \alpha p(x|y) P(y) \tag{5.15}$$

where $P(y)$ is the *a prior* probability of the label y, $p(y|x)$ is the *a posteriori* probability of the label y, $p(x|y)$ is the data distribution and α is a constant. Note that y takes two possible values: $0, 1$. The classifier should choose the the most probable label i.e., the label y which has the highest probability $P(y|x)$. This classifier is called the *Bayes classifier* and it is optimal in the sense that it has the smallest probability of error.

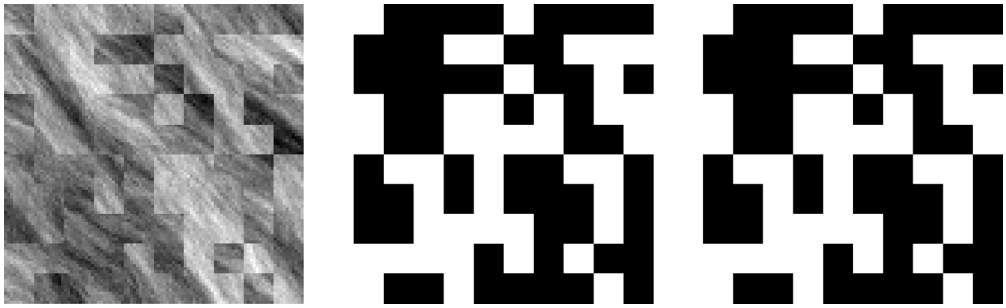


Figure 5.3: Texture classification: input image (left) true class (center) estimated class (right)

In many problems the probability distributions are unknown and all we know is a set of input-output pairs (x_i, y_i). In this case we have to learn the classifier and the story is much more complex. In fact there is a whole area in statistical learning which deals with this problem known as *supervised learning*. Some popular methods to classify data from a training set are: Fisher discriminant, the nearest neighbor classifier, classification trees, and support vector machines [3].

We will adopt a different approach. We will estimate the data distributions $p(x|y)$ and use the Bayes classifier. The data distribution are Gaussian $N(\mu_y, R_y), y \in \{0, 1\}$ where μ_y, R_y are the mean vector and covariance matrix which can be estimated from the training set by standard formulas

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad R = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^T \tag{5.16}$$

Using the data points in Figure 5.2 (right), we obtained

$$\mu_1 = \begin{bmatrix} 3.57 \\ 1.73 \end{bmatrix} \quad \mu_2 = \begin{bmatrix} 1.80 \\ 3.40 \end{bmatrix} \quad R_1 = \begin{bmatrix} 1.103 & 0.068 \\ 0.068 & 0.127 \end{bmatrix} \quad R_2 = \begin{bmatrix} 0.126 & 0.147 \\ 0.147 & 1.852 \end{bmatrix} \tag{5.17}$$

The most probable labels can now be computed using the Bayes law (5.15). The normal probability density function is

$$N(\mathbf{x}; \mu, \mathbf{R}) = \frac{1}{(2\pi)^{d/2}|\mathbf{R}|} e^{-\frac{1}{1}(\mathbf{x}-\mu)^T \mathbf{R}^{-1}(\mathbf{x}-\mu)} \tag{5.18}$$

where $d = 2$ is the dimension of $\mathbf{x}$ and $|\mathbf{R}|$ is the determinant of the covariance matrix. Figure 5.4 shows the test image made of two types of textures (left), the true label of the image blocks (center) and the estimated label (right).The estimated labels are identical to the true ones. No labeling error was made since the textures can be easily discriminated.

The Bayes classifier (known distributions) is summarized in table 5.3. The strategy followed in this problem (we used the expression of the Bayes classifier replacing the unknown parameters

Bayes Classifier

Goal: assign a label y to an observation x

Hypothesis: $p(x|y), P(x)$ are known

1. Compute the a posteriori probabilities: $P(y|x) = \alpha p(x|y)P(x)$
2. Choose the most probable label y

Table 5.2: Bayes classifier

by estimates) is suboptimal but it is a very common practice. This method is known as the plug-in Bayes classifier[2].

In the classification problem solved in this section we made some simplifying hypotheses. We will try to relax some of them in the next section.

5.4 Parameter estimation

We had to estimate the parameters of a multivariate Gaussian distribution from data in the previous examples. Fortunately, there are well known formulas for the mean and covariance matrix in this case and we did not bother to derive them. How should we proceed when we do not know expressions for the parameters estimates?

There are several well known methods to solve this problem. We will briefly review two of the most popular: the maximum likelihood methods proposed by Fisher and the MAP method.

Maximum Likelihood Method

Suppose we want to estimate a probability density function $p(x|\theta)$ of a random variable x where θ is an unknown parameter. In general x and θ are vectors. Let $X = \{x_i, i = 1, \ldots, n\}$ be a set independent realizations of x known as training set. We want to estimate the unknown parameter θ using the X.

The maximum likelihood (ML) method chooses the θ which makes the observations more probable i.e., which maximizes $p(X|\theta)$. The estimate is therefore defined by

$$\hat{\theta} = \arg\max_{\theta} L(\theta) \tag{5.19}$$

[2]the plug-in Bayes classifier is suboptimal because it neglects the uncertainty of the parameter estimates

where

$$L(\theta) = p(X|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \tag{5.20}$$

is known as the *Likelihood function.*

Sometimes we use the log likelihood function instead

$$\hat{\theta} = \arg\max_{\theta} l(\theta) \tag{5.21}$$

$$l(\theta) = \log p(X|\theta) = \sum_{i=1}^{n} \log p(x_i|\theta) \tag{5.22}$$

which is often simpler.

Example

Suppose we wish to estimate the parameter s of a Rayleigh distribution[3]

$$p(x) = \frac{x}{s^2} e^{-\frac{x^2}{2s^2}} \tag{5.23}$$

from a set of independent realization $X = \{x_i\}$.

The log likelihood function is given by

$$l(s) = \sum_{i=1}^{n} \frac{x_i}{s^2} e^{-\frac{x_i^2}{2s^2}} \tag{5.24}$$

Therefore,

$$l(s) = C - 2n \log s - \frac{n}{2s^2} S^2 \tag{5.25}$$

where

$$S^2 = \frac{1}{n} \sum_{i=1}^{n} x_i^2 \tag{5.26}$$

Computing the derivative of $l(s)$ and making it equal to zero we obtain

$$-\frac{2n}{f} + n\frac{n}{f^3} S^2 = 0 \tag{5.27}$$

$$\hat{s}^2 = \frac{1}{2} S^2 \tag{5.28}$$

This is the ML estimate of s.

---

[3]the Rayleigh distribution is used to describe the intensity of ultrasound images. In this case the parameter s varies along the image.

Maximum a Posteriori Method

The maximum *a posteriori* (MAP) method is a Bayesian method. It assumes that θ is itself a random variable and we know its distribution $p(\theta)$ also called the *a priori* distribution or simply *prior*.

After observing the data X we can update the a priori distribution of θ using the Bayes law and obtain

$$p(\theta|X) = \alpha p(X|\theta)p(\theta) \tag{5.29}$$

where $\alpha = p(X)$. This is the so-called a posteriori distribution. Note that there is a full equivalence with what we did before for the Bayes classifier. Everything is the same except that we now wish to estimate a parameter (integer or a vector) instead of a class label.

The MAP method chooses the most probable value of θ using the a posteriori distribution

$$\hat{\theta} = \arg\max p(\theta|X) \tag{5.30}$$

$$\hat{\theta} = \arg\max p(X|\theta)p(\theta) \tag{5.31}$$

We note the similarity with the ML method. The MAP method maximizes the ML function multiplied by the prior. The prior is the new term which accounts for the existing information about the parameter θ before the observations.

This is also a sound framework to perform data fusion if we have multiple sources of information or information obtained at different instants of time. The estimate available at time $t-1$ can be updated with new information X^t by

$$\hat{\theta}^t = \arg\max_{\theta} p(X^t|\theta)p_{t-1}(\theta) \tag{5.32}$$

where

$$p_{t-1}(\theta) = p(\theta|X^1, \ldots, X^{t-1}) \tag{5.33}$$

is the distribution of θ with all the information available until time $t-1$ and updated by

$$p_t(\theta) = \alpha_t p(X^t|\theta)p_{t-1}(\theta) \tag{5.34}$$

Example Suppose we want to estimate the mean of a normal distribution $N(\mu, \sigma^2)$ with known σ known that the prior distribution of μ is normal $N(\mu_0, \sigma_0^2)$.

The log likelihood is

$$l(\mu) = \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2\sigma^2}(x_i-\mu)^2} \tag{5.35}$$

$$l(\mu) = -n \log \sqrt{2\pi\sigma^2} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 \tag{5.36}$$

The logarithm of the prior is,

$$\log p(\mu) = -\frac{1}{2} \log 2\pi\sigma_0^2 - \frac{1}{2\sigma_0^2}(\mu - \mu_0)^2 \tag{5.37}$$

The function to be maximized is

$$l(\mu) + \log p(\mu) = C - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 - \frac{1}{2\sigma_0^2}(\mu - \mu_0)^2 \tag{5.38}$$

Computing the derivative with respect to μ we obtain,

$$\frac{1}{\sigma^2} \sum_{i=1}^{n} (\mu - x_i) + \frac{1}{\sigma_0^2}(\mu - \mu_0) = 0 \tag{5.39}$$

$$\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2} \right) \mu = \frac{n\bar{x}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2} \tag{5.40}$$

$$\hat{\mu} = \frac{\bar{x} + \frac{\sigma^2 \mu_0}{n\sigma_0^2}}{1 + \frac{\sigma^2}{n\sigma_0^2}} \tag{5.41}$$

where $\bar{x}$ is the average value of x in the training set. The estimate obtained with the MAP method depends on the data and on the prior. The influence of the prior is negligeable if n is large. However if we have few data points the prior plays an important role.

5.5 Unsupervised Texture Classification

Suppose we had a textured image to learn the classifier but we did not know the true label of each block. The Data used to learn the classifier only has the input x but does not include the desired output (label) y. The training set is therefore given by $\{x_i\}$. How do we estimate the relationship between x and y if we do not have any examples of y.

This seems an impossible problem since we know nothing about the input-output (x, y) relationship.

It is amazing but we can still solve this problem in an efficient way and achieve good results if the two clouds do not overlap too mutch. Suppose we have the same training data as before but without the labels i.e., we have the data points shown in Fig. 5.2(right) but they do not have the color labels since the label (color) is unknown. We can still separate the two clouds and estimate the mean vector and covariance. This seems intuitive if we look at the figure and look for ellipsoidal clouds of points. However, how do we do this in an automatic way?

The probability distribution is a mixture of two Gaussians

$$p(y) = c_0 N(x; \mu_0, R_0) + c_1 N(x; \mu_1, R_1) \tag{5.42}$$

$$N(x; \mu, R) = \frac{1}{2\pi |R|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T R^{-1}(x-\mu)} \tag{5.43}$$

where $c_0, c_1 (c_0 + c_1 = 1)$ are the mixing probabilities and $\mu_i, R_i, i = 0, 1$ are the parameters of the Gaussians. How can we estimate the mixture parameters?

This problem is solved by the Expectation-Maximization (EM) method. This method allows us to fit the data set $\{x_i\}$ by two Gaussian distributions. The idea is simple. First we initialize the mean and covariance of the two Gaussians (μ_1, μ_2, R_1, R_2). Then we compute the probability of both labels for each data point i.e., we compute the weights

$$w_{ij} = P(y_i | x_i) = \alpha p(x_i | y = j) P(y = j) \tag{5.44}$$

The weights sum one $(w_{i0} + w_{i1} = 1)$ and they have an intuitive meaning: w_{ij} is the degree of membership of x_i to label j. After this step, we update the Gaussian parameters using all the data points wighted by their associated probabilities as follows

$$\mu_j = \frac{\sum_{i=1}^N w_{ij} x_i}{\sum_{i=1}^N w_{ij}} \qquad R_j = \frac{\sum_{i=1}^N w_{ij} (x_i - \mu)(x_i - \mu_j)^T}{\sum_{i=1}^N w_{ij}} \tag{5.45}$$

Let us apply the EM method to the data points of Figure 5.2 (right) without labels (colors). The mean vectors μ_1, μ_2 were initialized by randomly selecting two data vectors and the covariance matrices R_1, R_2 were made equal to the identity matrix. Figure 5.4 shows the results obtained at iterations 0 (initialization), $1, 5$ and 20. It shows level curves of the probability distribution which are ellipses and the weights associated to the label 0. Initially the weights are close to 0.5 for most of the data points because the covariance matrices are very large and all data vectors can be represented by both Gaussians. As the estimates improve the weight vectors become closer either to 0 or 1 as expected.

After estimating the probabilistic model using the EM method we applied this model to classify the texture blocks of the test image 5.4. We have obtained a perfect classification as

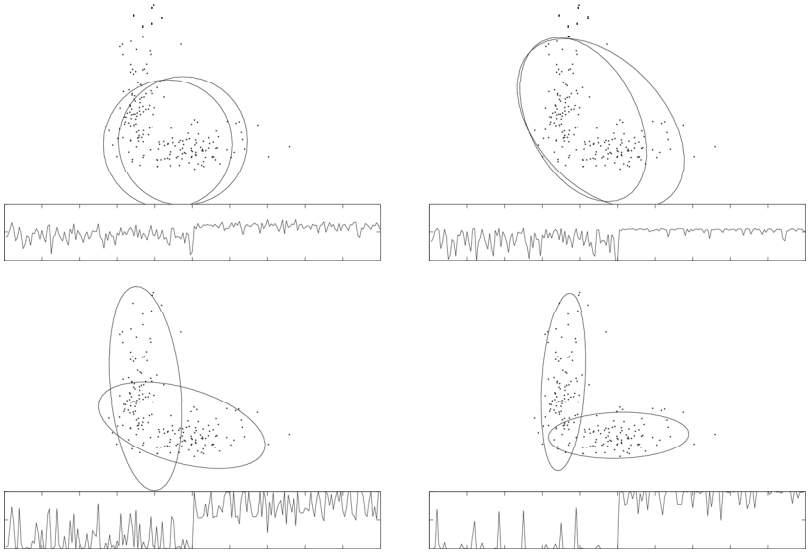


Figure 5.4: Mixture estimation with the EM method: data points, level curves and degree of membership w_{0j} at iterations 0 (random initialization), $1, 5, 20$.

before. The EM method for a mixture of Gaussians is summarized in table 5.3. The method can be used in under more general conditions [3].

5.6 Dealing with Arbitrary Shapes

We discussed how two label different textures by learning a function $y = f(x; \theta)$ when we have a labeled training set (supervised learning) and when we do not known the correct labels in the training set (unsupervised learning). We have assumed that the image consisted of non overlapping blocks of homogeneous texture.

What should we do when this hypothesis is not valid and the labeled regions have arbitrary shapes? This is a difficult step. First feature extraction becomes more difficult since it depends on the labeling step. We can still compute the mean of the intensity differences in each region but the regions shapes depend of the labeling.

Second we need a label for each pixel instead of assigning a common label to all the pixels of a block. Region shape influences feature extraction and vice versa.

We will discuss this type of problem latter using random space models (Markov random fields).

EM method

Goal: estimate the parameters of a Gaussian mixture $p(x) = \sum_{j=1}^{m} c_j N(\mu_j, R_j)$

Data: $\{x_i\}$

initial

iterate:

E step: compute the label probabilities for each data point

$$w_{ij} = \alpha_i N(x; \mu_j, R_j) c_j \qquad j = 1, \ldots, m \tag{5.46}$$

M step: update the mixture parameters (c_j, μ_j, R_j)

$$c_j = \frac{1}{N} \sum_{i=1}^{N} w_{ij} \qquad \mu_j = \frac{\sum_{i=1}^{N} w_{ij} x_i}{\sum_{i=1}^{N} w_{ij}} \qquad R_j = \frac{\sum_{i=1}^{N} w_{ij} (x_i - \mu)(x_i - \mu_j)^T}{\sum_{i=1}^{N} w_{ij}} \tag{5.47}$$

end

Table 5.3: EM Algorithm: estimation of mixtures of Gaussians

Bibliography

[1] A. K. Jain , R. P.W. Duin , J. Mao, Statistical Pattern Recognition: A Review, IEEE Transactions Pattern Analysis and Machine Intelligence, Vol. 22, No. 1, pp. 4-37, Janeiro 2000.

[2] J. Marques, Reconhecimento de Padrões, Métodos Estatísticos e Neuronais, 1999.

[3] R. Duda, P. Hart, D. Stork, Pattern Classification, Wiley, 2001.

[4] T. Hastie, R. Tibshirani, J. Friedman, The Elements os Statistical Learning, Data Mining, Inference and Prediction, Springer, 2001.

[5] P. Viola, M. Jones, Robust Real-time Object Detection, 2nd International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing and Sampling, 2001.

Part II

Image Processing

Chapter 6

Image Enhancement and Restoration

6.1 Introduction

Images are often degraded during the acquisition process (camera out of focus, moving camera, lost image pixels, noise). The goal of image enhancement and restoration is to recover from these errors as much as possible.

To perform this operation we need to make hypothesis about the image properties and about the type of degradation that occurred. These assumptions may be specified by the user or learned from the data. For example, if the observed image is corrupted by noise we can either specify the noise statistics (e.g., white Gaussian noise) or learn the noise model from original and distorted images.

This topic is an excellent introduction to model based image processing since it combines many important issues: image models, acquisition models and inference techniques. The methods used in this section are useful to address other image processing problems such as image alignment and super resolution.

Figure 6.1: Degradation with white Gaussian noise

6.2 Inverse problems

Image restoration is an example of a wider class of problems called *inverse problems*. These problems aim to invert the transformation

$$\mathbf{y} = \mathcal{A}\mathbf{x} \tag{6.1}$$

where $\mathbf{y}$ is the observed variable, $\mathbf{x}$ is the unknown variable and $\mathcal{A}$ is a non invertible operator or an ill-conditioned operator. The above equation does not have a unique solution.

Let us consider the equation

$$y = x_1 + x_2 \tag{6.2}$$

where x_1, x_2 are two unknowns. It is not easy to determine them from their sum y.

This chapter gives several methods to address this type of problems.

6.3 Degradation Model

We will assume that there is an original image (unknown) which is distorted in some way. For example, let us assume that the original image $I(\mathbf{x})$ is corrupted by white Gaussian noise. The observed image is therefore

$$J(\mathbf{x}) = I(\mathbf{x}) + W(x) \tag{6.3}$$

where $W(\mathbf{x})$ is a noise image i.e., $E\{\mathbf{W}(\mathbf{x})\mathbf{W}(\mathbf{x}')\} = \sigma^2\delta(\mathbf{x}, \mathbf{x}'), \mathbf{W}(\mathbf{x}) \sim \mathcal{N}(0, \sigma^2)$. Figure 6.1 shows an example of this degradation process.

The acquisition process is often more complex involving two types of degradation: a deterministic degradation (e.g., burr) and a random degradation. In this case we can often assume that

$$J = \mathcal{T}(I) + W \tag{6.4}$$

where $\mathcal{T}$ is a deterministic operator and W is the observation noise. In some problems we may assume that $\mathcal{T}$ is a linear operator. In this case, J is the convolution of the original image I with an impulse response:

$$\mathcal{T}(I)(\mathbf{x}) = \sum_{\mathbf{x}'} h(\mathbf{x}, \mathbf{x}')I(\mathbf{x}') \qquad \text{space-varying} \tag{6.5}$$

or

$$\mathcal{T}(I)(\mathbf{x}) = \sum_{x'} h(\mathbf{x} - \mathbf{x}')I(\mathbf{x}') \qquad \text{space-invariant} \tag{6.6}$$

For example, out of focus picture and motion blur can be modeled in this way using appropriate degradation filters. In the first case the impulse response is isotropic while in the second the impulse response depends on the motion direction.

It is often useful to represent the original and observed images with vector notation. Let $\mathbf{I}, \mathbf{J}$ be the vectors obtained by concatenation of the columns if the original and degraded images; the dimension of these vectors is $n \times 1$ where n is the number of pixels of the image[1].

The linear degradation process can now be stated as

$$\mathbf{J} = \mathbf{M}\mathbf{I} + \mathbf{W} \tag{6.7}$$

where $\mathbf{M}$ is a degradation matrix and $\mathbf{W}$ a random matrix. This is a very useful notation. However, it should not be used to simulate image degradation in a computer since the dimension on $\mathbf{M}$ is very high (e.g., $10^6 \times 10^6$).

6.4 The naive approach

To estimate $\mathbf{I}$ from $\mathbf{J}$ we must be able to separate $\mathbf{MI}$ from $\mathbf{W}$ in (6.7). This is only possible if we make some hypotheses about both terms.

Let us ignore this statement for the moment and try to apply a classic estimation method (maximum likelihood) in a blind way.

Assuming that the noise image $\mathbf{W}$ is white and Gaussian the log likelihood function defined by

$$l(I) = \log p(J|I) \tag{6.8}$$

can be easily computed

$$l(I) = \log \prod_{\mathbf{x}} p(J(\mathbf{x})|I(\mathbf{x})) = \sum_{\mathbf{x}} \log p(J(\mathbf{x})|I(\mathbf{x})) \tag{6.9}$$

[1] the symbol $\mathbf{I}$ is being used with two meanings: original image and identity matrix; the meaning should be clear from the context

Since the noise is Gaussian $p(J(x)|I(x) = N(J(x); I(x), \sigma^2))$

$$l(I) = -\frac{n}{2}\log(2\pi\sigma^2) - \sum_{\mathbf{x}} \frac{1}{2\sigma^2}(J(\mathbf{x}) - I(\mathbf{x}))^2 \tag{6.10}$$

Function l is maximized if $\hat{I}(\mathbf{x}) = J(\mathbf{x})$. This means that the ML estimate of I is the observed image$J$. Discouraging. According to this method we should not modify the observed image. What is missing? The maximum likelihood model makes no assumption about one of the signals: the original image. We need to use a model for the image as well.

In image restoration problems there is often another difficulty **.** There is also loss of information when we apply the operator T. The operator is non invertible is many problems.

Example

Let us consider the following problem: $\mathbf{I} = (I_1, I_2)$ is an unknown vector and we only observe the sum of both components ($J = I_1 + I_2$). There is also loss of information in this case. How can we estimate $\mathbf{I}$ from J?

We must make some assumptions about the solution.

6.5 Regularization Methods

The previous difficulties can be overcome by assuming that the reconstructed signal I has to verify additional restrictions. This raises an important question: how can we characterize the class of admissible images. The answer to this question has a direct influence on the solution.

In this section we will assume that the image I is smooth. We wish to find an image I such that $\mathcal{T}(I) \approx J$ i.e., the (squared) norm of the reconstruction error

$$\|J - \mathcal{T}(I)\|^2 \tag{6.11}$$

is small. However this is not enough to obtain an unique solution and we wish to impose an additional constraint. We wish to find the smoother image among all the images with small reconstruction error. The smoothness criterion is defined as

$$\|D(I)\|^2 \tag{6.12}$$

where D is a differential operator, e.g., an operator which computes difference between pairs of neighboring pixels[2].

These two different goals can be included in a single cost functional

$$C = \|J - \mathcal{T}(I)\|^2 + \alpha \|D(I)\|^2 \tag{6.13}$$

with two terms: a data term which measures the fitness of the model to the observed data and a regularization term; α controls the relative importance of both criteria. The minimization of C leads to a reconstructed image which is compatible with the observations and it is smooth. The methods based on the minimization of (6.13) are called *regularization methods*.

These principles can be applied to the denoising problem using the Euclidean norm. In this case,

$$C = \sum_{\mathbf{x}} (J(\mathbf{x}) - I(\mathbf{x}))^2 + \alpha \sum_{(\mathbf{x},\mathbf{y}) \in V} (I(\mathbf{x}) - I(\mathbf{y}))^2 \tag{6.14}$$

where V the set of all neighboring pixels.

To minimize C, all the partial derivatives with respect to $I(\mathbf{x})$ must be zero

$$\frac{\delta C}{\delta I(\mathbf{x})} = 0 \tag{6.15}$$

Therefore,

$$-2(J(\mathbf{x}) - I(\mathbf{x})) + \alpha \sum_{\mathbf{y} \in V_{\mathbf{x}}} 2(I(\mathbf{x}) - I(\mathbf{y})) = 0 \tag{6.16}$$

where $V_{\mathbf{x}}$ is the set of neighbors of pixel $\mathbf{x}$. Therefore, we obtained a system of linear equations

$$(1 + \alpha N_{\mathbf{x}}) I(\mathbf{x}) - \alpha \sum_{\mathbf{y} \in V_{\mathbf{x}}} I(\mathbf{y}) = J(\mathbf{x}) \tag{6.17}$$

where $N_{\mathbf{x}}$ is the number of neighbors of pixel $\mathbf{x}$. The previous equations can be recursively solved.

$$I(\mathbf{x}) = \frac{J(\mathbf{x}) + \alpha N_{\mathbf{x}} \bar{I}(\mathbf{x})}{(1 + \alpha N_{\mathbf{x}})} \tag{6.18}$$

where $\bar{I}(\mathbf{x})$ is the average intensity of all the neighbors of pixel $\mathbf{x}$ (excluding $\mathbf{x}$), computed using the last estimate of the image. Pixel update may be performed using two different strategies. We may update each pixel as soon as we have a new estimate for it. In this case we are solving the system of equations using the *Gauss-Seidel method*. Instead, we may compute all pixel estimates first and update all the pixels simultaneously. In this case we are using the *Jacobi method*.

[2] we note that the two norms are defined in different spaces and can be independently chosen.

Matrix notation is very useful to describe the regularization method with quadratic cost even if some of the matrices can not be stored on a computer due to their huge dimensions. Let $\mathbf{I}, \mathbf{J}$ be two column vectors obtained by stacking the columns of the corresponding images. The degradation model can be written as $\mathbf{J} = \mathbf{I} + \mathbf{W}$ where $\mathbf{W}$ is a noise vector. The cost functional can now be written as

$$C = \|\mathbf{J} - \mathbf{I}\|^2 + \alpha \|\mathbf{D}\mathbf{I}\|^2 \tag{6.19}$$

$$C = (\mathbf{J} - \mathbf{I})^T (\mathbf{J} - \mathbf{I}) + \alpha \mathbf{I}^T \mathbf{D}^T \mathbf{D}\mathbf{I} \tag{6.20}$$

where $\|.\|$ stands for the Euclidean norm and $\mathbf{D}$ is a sparse matrix which computes the differences between pairs of neighboring pixels. Computing the derivative of C with respect to vector $\mathbf{I}$ we obtain[3]

$$-(\mathbf{J} - \mathbf{I}) + \alpha \mathbf{D}^T \mathbf{D}\mathbf{I} = 0\,. \tag{6.21}$$

and

$$(\mathbf{I} + \alpha \mathbf{D}^T \mathbf{D})\mathbf{I} = \mathbf{J} \tag{6.22}$$

This is a linear system of equations written with matrix notation.

Although this equations is simple it is not easy to invert matrix $\mathbf{I} + \alpha \mathbf{D}^T \mathbf{D}$ since it has a huge dimension $n \times n$ where n is the number of pixels in the image. In fact we cannot even store it in a computer. In practice, recursive algorithms are used to solve the system of equations such as the ones described before.

Figure 6.2 (top-right) shows the cameraman figure corrupted by white noise with SNR=13dB. It also shows the reconstruction results obtained with the regularization method using $\alpha = 1$ (SNR=17.0dB) and $\alpha = 8$(SNR=18.6dB). The first choice of α has corresponds to a mild regularization effect. The input noise is still clearly visible at the output. The second choice for α corresponds to a strong regularization which attenuates the noise but it also blurs the image. The final result looks as if the image was out of focus. This is the trade off involved in the choice of α.

6.6 Edge preserving methods

The previous method attenuates the noise by performing a low pass filtering of the signal. Therefore, fast transitions are filtered and replaced by smooth transitions. This creates a *burr* effect which is perceptually annoying since it destroys the edge information.

[3]see the derivative rules in Appendix

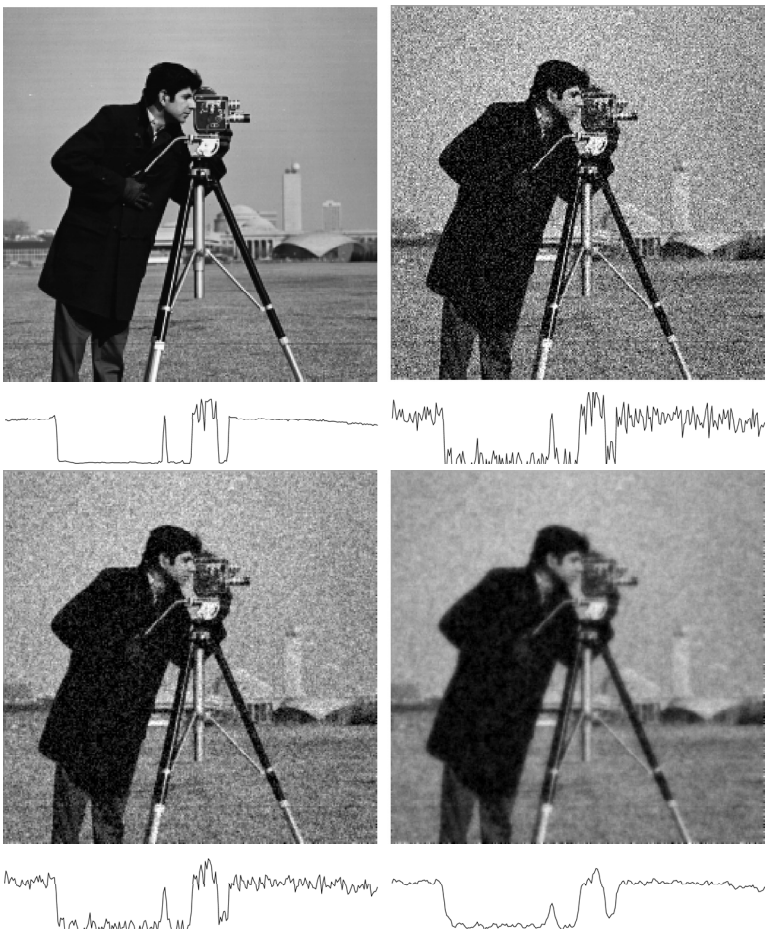


Figure 6.2: Denoising with Euclidean norms (clockwise): original image and intensity profile (line 100), noisy image (SNR=13dB) and profile, reconstruction results with $\alpha = 1$ (SNR=17.6dB) and $\alpha = 8$ (SNR=18.5dB)

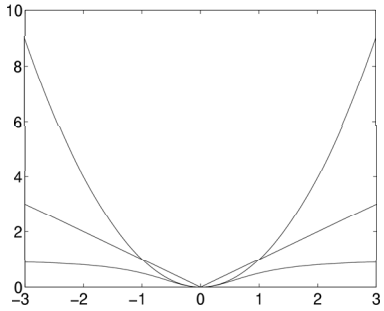


Figure 6.3: Cost functions: $x^2, |x|$ and $x^2/(1+x^2)$ (x and y scales are different)

This raised an important question in image reconstruction: *can we attenuate the noise without destroying the edges?*

The bad performance of the previous method at the transitions is a consequence of the way transitions are penalized by the cost function C. The squared Euclidean distance $\|I(\mathbf{x})-I(\mathbf{y})\|^2$ rapidly grows when there is a sudden change of intensity in image I. It would be better if we the transition cost grew slower (see figure 6.3) in such a way that small changes due to noise would still be penalized but very large changes due to transitions would not receive a very high penalty.

One way to reduce the cost of abrupt changes consists of using the l_1 norm in the regularization term

$$C = \sum_{\mathbf{x}} (J(\mathbf{x}) - I(\mathbf{x}))^2 + \alpha \sum_{(\mathbf{x},\mathbf{y})\in V} |I(\mathbf{x}) - I(\mathbf{y})| \tag{6.23}$$

This norm grows slower than the l_2 norm and it is less sensitive to abrupt changes. However, the minimization of C becomes a nonlinear problem.

This difficulty can be partially overcome by converting this problem into a sequence of quadratic problems which can be solved by linear methods. Equation (6.23) can be written as

$$C = \sum_{\mathbf{x}} (J(\mathbf{x}) - I(\mathbf{x}))^2 + \alpha \sum_{(\mathbf{x},\mathbf{y})\in V} w(\mathbf{x},\mathbf{y})(I(\mathbf{x}) - I(\mathbf{y}))^2 \tag{6.24}$$

where

$$w(\mathbf{x},\mathbf{y}) = \frac{1}{|I(\mathbf{x}) - I(\mathbf{y})| + \epsilon} \tag{6.25}$$

is a weight and ϵ is a small constant used to prevent the weights from tending to infinity. The minimization of (6.24) is similar to the minimization of (6.23) if we consider the dependence of the weights $w(\mathbf{x},\mathbf{y})$ on the image $I(\mathbf{x}) - I(\mathbf{y})$. However, if the weights are kept constant in each iteration (simplifying hypothesis) the problem becomes mutch simpler since the cost functional

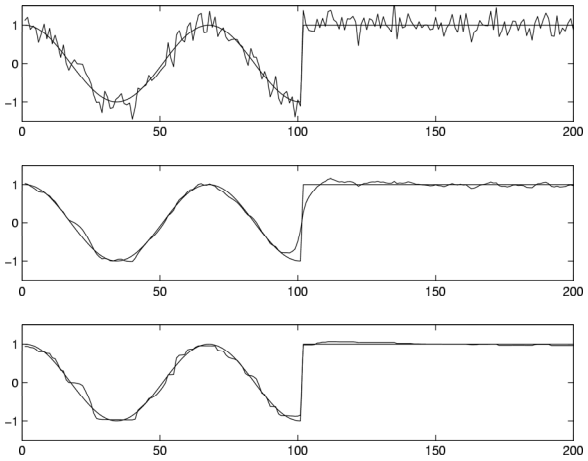


Figure 6.4: Denoising with the regularization method: (a) original and observed signals; restored signals with (b) squared Euclidean norm (SNR=17.4 dB) and with (c) the l_1 norm (SNR=21.7 dB)

becomes quadratic and can be solved as follows

$$[1 + \alpha \sum_{\mathbf{y} \in V_{\mathbf{x}}} w(\mathbf{x}, \mathbf{y})]\, I(\mathbf{x}) = J(\mathbf{x}) + \alpha \sum_{\mathbf{y} \in V_{\mathbf{x}}} w(\mathbf{x}, \mathbf{y}) I(\mathbf{y}) \tag{6.26}$$

$$I(\mathbf{x}) = \frac{J(\mathbf{x}) + \alpha \sum_{\mathbf{y} \in V_{\mathbf{x}}} w(\mathbf{x}, \mathbf{y}) I(\mathbf{y})}{1 + \alpha \sum_{\mathbf{y} \in V_{\mathbf{x}}} w(\mathbf{x}, \mathbf{y})} \tag{6.27}$$

Since the weights depend on the estimate I they must be updated at the end of each iteration and the cost functional must be minimized again. The process usually converges after a few iterations.

Figure 6.4 illustrates the performance of the regularization method at transitions using the norms l_2 and l_1. The best results at transitions are achieved with the norm l_1 while the l_2 norm achieves slightly better results at smooth regions. It is also observed that the l_1 norm tends to produce a piecewise constant estimate i.e., it is very well adapted to that class of signals. The output of this algorithm in the cameraman example is shown in Figure 6.5.

6.7 Bayesian Methods

The Bayesian methods formulate the reconstruction problem in a different way but they lead to similar algorithms as we shall see in this section.

Bayesian methods assume that the unknown image I is a realization of a random variable

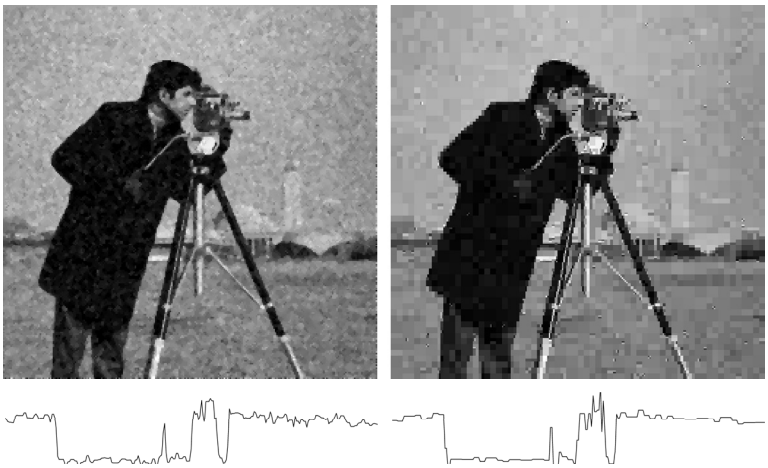


Figure 6.5: Denoising with the norm L_1 (left, SNR=19.3dB) and using wavelets with soft thresholding (right, SNR=18.5dB)

with a known distribution $p(\mathbf{I})$ called the *prior*. The prior contains all the available information about the statistical properties of the image e.g., that it is a smooth signal in most of the points and may exhibit sudden changes at a small subset of points.

Bayesian methods compute the *a posteriori* distribution of $\mathbf{I}$ after observing the noisy image $\mathbf{J}$. The *a posteriori* distribution is obtained using the Bayes law

$$p(\mathbf{I}|\mathbf{J}) = kp(\mathbf{J}|\mathbf{I})p(\mathbf{I}) \tag{6.28}$$

where $p(\mathbf{J}|\mathbf{I})$ is called the sensor model and describes the degradation process, $p(\mathbf{I})$ is the *prior* and k is a normalization constant.

To estimate $\mathbf{I}$ we can compute the maximum of the *a posteriori* distribution $p(\mathbf{I}|\mathbf{J})$. This is the most probable value for the unknown image and the method is known as the maximum *a posteriori* (MAP) method

$$\hat{\mathbf{I}} = \arg\max_{\mathbf{I}} p(\mathbf{I}|\mathbf{J}) \tag{6.29}$$

The MAP method requires the specification of $p(\mathbf{J}|\mathbf{I})$ (sensor model) and $p(\mathbf{I})$ *prior* distribution for the image. The method may lead to simple or complex *a posteriori* distributions depending on the sensor model and *prior*. A simple case is obtained if the image is corrupted by white Gaussian noise $\mathbf{W} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, and the *prior* is Gaussian as well $\mathbf{I} \sim \mathcal{N}(\mathbf{0}, (\alpha\mathbf{D}^T\mathbf{D})^{-1})$. In this case

$$p(\mathbf{J}|\mathbf{I}) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp(-\frac{1}{2\sigma^2}(\mathbf{J}-\mathbf{I})^T(\mathbf{J}-\mathbf{I})) \tag{6.30}$$

$$p(\mathbf{I}) = \frac{\det(\alpha \mathbf{D}^T \mathbf{D})}{(2\pi)^{d/2}} \exp(-\frac{\alpha}{2} \mathbf{I}^T D^T D \mathbf{I}) \tag{6.31}$$

The image estimate $\mathbf{I}$ is obtained maximizing

$$p(\mathbf{J}|\mathbf{I}) p(\mathbf{I}) = K \exp(-\frac{1}{2\sigma^2} (\mathbf{J} - \mathbf{I})^T (\mathbf{J} - \mathbf{I}) - \frac{\alpha}{2} \mathbf{I}^T \mathbf{D}^T \mathbf{D} \mathbf{I}) \tag{6.32}$$

This is equivalent to the minimization of

$$C = (\mathbf{J} - \mathbf{I})^T (\mathbf{J} - \mathbf{I}) - \sigma^2 \alpha \mathbf{I}^T \mathbf{D}^T \mathbf{D} \mathbf{I} \tag{6.33}$$

which is the same as the regularization cost functional (6.20) with Euclidean norms. Both methods become identical if the regularization term is based on the Euclidean norm and the prior is Gaussian *prior* $N(\mathbf{0}, (\alpha \mathbf{D}\mathbf{D}^T)^{-1})$. The prior plays the role of the regularization term.

6.7.1 Denoising with wavelets

Explicar o soft e hard thresholding.

6.8 Moisaicing and Superresolution

Exercises

- Extend the regularization method for the linear model $\mathbf{J} = \mathbf{H}\mathbf{I} + \mathbf{W}$, where H is a square image.

Chapter 7

Image Segmentation

7.1 Introduction

Image segmentation is a basic image operation. It amounts to finding regions associated with the objects of interest, in the image domain. Our visual system performs this task in a natural way. We have no difficulty to distinguish all sorts of objects when we look at a scene. However, segmentation is one of the most challenging operations in image analysis.

Figure 7.1 shows some examples from different application areas. Segmentation is used in document image analysis to separate written text from the background, allowing the application of recognition modules. In remote sensing, image segmentation is used for a variety of problems e.g., to classify the terrain cover as forest, agriculture, urban, water. Image segmentation is also a key operation in medical applications. It is used to extract the boundary of organs and anatomic structures in images obtained by different types of modalities (CAT, MRI, US). A fourth example is drawn from the video surveillance area. Modern video surveillance systems must be able to segment moving objects in an image (e.g., vehicles in highways) in order to classify them as well as their activities (e.g., classify dangerous maneuvers is a highway).

Image segmentation is a difficult task. It is not easy to include the a priori information we have about the object (shape, color) in an objective criterion. Most algorithms are still far from that perspective and are based on two simple ideas:

- **region homogeneity:** the image should be split into *homogeneous regions* with similar intensity, color, texture or motion
- **transitions:** transitions (e.g., edge points) detected in the images are used to estimate

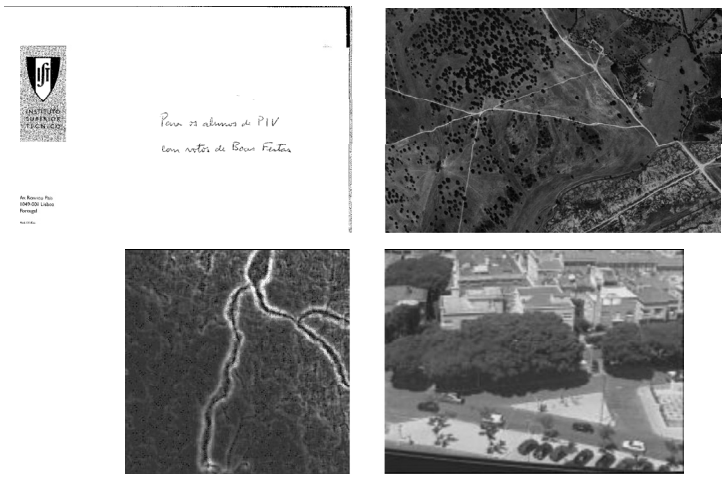


Figure 7.1: Segmentation problems

the object boundaries.

The first type of methods estimate regions using an homogeneity criterion while the second ones estimate the objects boundaries. There is an underlying assumption that objects are associated homogeneous regions with abrupt transitions at their boundary. This is not true in many cases. Therefore the performance of most methods is still limited.

Prior information about the scene plays a very important role. If we are segmenting cows in a green field we should take this information into account. Therefore segmentation is not a pure bottom up problem. It should take into account the available information about the objects present in the scene (top-down). This is a challenging problem because it is not easy to represent the objects shape, color and texture information in useful way. How can we teach a computer to segment people in a mob?

The next sections formulate the segmentation problem and present some of the available techniques from simple thresholding methods to Markov random fields or level set approaches.

7.2 Problem Formulation

Given an image u we wish to split the image domain into n regions $S_k, k = 0, \ldots, L-1$ where S_0 is the background image and the other regions are associated with different objects of interest (Figure 7.2). We will assume that $\{S_k\}$ is a partition of the image domain i.e., all S_k are disjoint and their union is the whole domain.

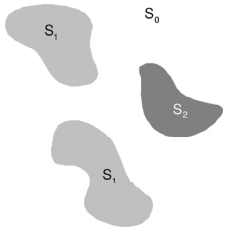


Figure 7.2: Homogeneous regions

This operation can be performed by assigning a label

$$l_k \in \{0, 1, \ldots, n-1\} \tag{7.1}$$

to each pixel $(k, u(k))$. The label sequence $l = (l_0, \ldots, l_{L-1})$ is equivalent to the image partition into disjoint regions $\{X_k\}$.

The human visual system performs visual segmentation using many types of cues and a great amount of knowledge about the visual appearance of different types of objects in different poses. This is a formidable task which can not be performed by current image analysis systems.

Most of the approaches described bellow are based on some kind of model or homogeinity criterion which is assumed to be valid to represent the image in each region S_k. For each specific application we must be able to choose the most appropriate method.

7.3 Thresholding

Consider the image u shown in Figure 7.1a. The text is darker than the background. One way to separate the text from the background consists of comparing the intensity of each pixel with a threshold T and assigning the label 1 if the intensity is smaller than the threshold. Therefore,

$$l(\mathbf{k}) = \begin{cases} 1 & I(\mathbf{k}) < T \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

Figure 7.3 shows the binary segmentation obtained with this method. This technique is called *thresholding* and it is used when we need a fast binary segmentation under simple conditions (the two classes are separable in the feature domain).

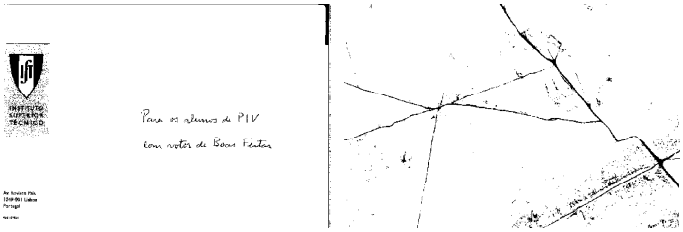


Figure 7.3: Binary segmentation obtained by thresholding

7.4 Pixel Classification

We can extend these ideas to L-ary decision problems. Suppose we wish to separate an image I into L regions and we can assume that the pixels inside each region S_i are independent realizations of a random variable with probability distribution $p(I(\mathbf{k})|\theta_i)$ where θ_i characterizes the color or intensity of the i-th region (see Figure 7.2).

Since the pixels are independent we can classify each of them in an independent way by choosing the most probable label l_k which maximizes the class probability

$$P(i|I(\mathbf{k})) = \alpha p(I(\mathbf{k})|\theta_i)P_i \tag{7.3}$$

where P_i is the *a priori* probability of the $i-th$ class and $\alpha = 1/p(I(\mathbf{k}))$ is a normalization constant which does not influence the decision.

Instead, we could assign a cost to each label as follows

$$E_{\mathbf{k}}(l) = -\log P(I(\mathbf{k})|\theta_l) - \log P_l \tag{7.4}$$

and choose the label with smallest cost.

The key point here is how to obtain the models $p(I|\theta_i)$ associated to the different objects (regions). A similar problem was discussed in chapter 5.

We can distinguish two approaches. In the first approach the user is asked to identify the regions S_i associated to each class of objects. In this case we have a supervised learning problem. The classified images provided by the user allow us to estimate the unknown parameters θ_l using standard estimation algorithms e.g., the ML method

$$\hat{\theta_i} = \arg\max_{\theta_i} p(I_i|\theta_i) \tag{7.5}$$

$$\hat{\theta_i} = \arg\max_{\theta_i} \prod_{k \in X_i} p(I(k)|\theta_i) \tag{7.6}$$

where $I_i = \{I(\mathbf{k}), \mathbf{k} \in X_i\}$. Suppose we want to classify the land cover using satellite images. First we would select patches from all the classes (e.g., water, agriculture, urban areas) to be separated. Then we would use learning methods to estimate the statistical models and classify the other pixels. This procedure can be applied to many problems (e.g., motion detection, pedestrian detection, face detection).

When we do not have classified data the learning problem becomes much more difficult since the image contains L objects but we do not know where they are. This is an *unsupervised learning problem* which can be solved using clustering algorithms [3]. The distribution of the data is a mixture

$$p(I(\mathbf{k})|\mathbf{c}, \theta) = \sum_{i=0}^{L-1} c_i p(I(\mathbf{k})|\theta_\mathbf{i}) \tag{7.7}$$

where $\mathbf{c} = (c_0, \ldots, c_L - 1)$ are the mixture coefficients and $\theta = (\theta_\mathbf{0}, \ldots, \theta_{\mathbf{L-1}})$ are the parameters of the mixture modes. The mixture coefficients are the *a priori* probabilities of the classes. Therefore $c_k \geq 0$ and

$$\sum_{k=0}^{L-1} c_k = 1 \tag{7.8}$$

The mixture mode $p(I(\mathbf{k})|\theta_\mathbf{i})$ is the distributions of the data associated to the i-th class. The estimation of $\mathbf{c}, \theta$ is much more difficult than in the supervised case because we do not know what is the class of each observation $I(\mathbf{k})$.

The Expectation-Maximization (EM) algorithm is a very useful technique to estimate the mixture parameters. The EM method is an iterative algorithm with 2 steps in each iteration. In the E-step, all the data points are classified in one of the available classes. This is done using a soft classifier i.e., by computing the probability of each class

$$w_{\mathbf{k}i} = P(l_\mathbf{k} = i|I(\mathbf{k}), \hat{\theta})) \tag{7.9}$$

$$w_{\mathbf{k}i} = \alpha p(I(\mathbf{k})|\hat{\theta}_i)\hat{c}_i \tag{7.10}$$

using the current parameters $\hat{\theta}, \hat{\mathbf{c}}$. The parameters of the mixture are updated using available weights computer by the EM method (see Chapter 5).

Background subtraction

Figure ? shows the image obtained with a surveillance camera and the evolution of a the intensity of a given pixel during a time interval. Most of the time the intensity is close to an average intensity $\bar{I}(\mathbf{k})$ plus some additional noise. Therefore,

$$I(\mathbf{k}, t) = \bar{I}(\mathbf{k}) + w(\mathbf{k}, t) \tag{7.11}$$

Figure 7.4: Motion detection by background subtraction [2]. Active regions are represented by the corresponding bounding boxes

where $w(\mathbf{k}, t)$ is a white Gaussian noise with zero mean and variance $\sigma(\mathbf{k})^2$. Therefore, $w(\mathbf{k}, t) \sim N(0, \sigma(\mathbf{k})^2)$. The mean and variance of each pixel can be computed from a video sequence using robust estimation methods. The image $\bar{I}$ is often called the *background image*.

Using this model, we can detection motion by comparing the absolute error $|I(\mathbf{k}, t) - \bar{I}(\mathbf{k})|$ with a known threshold. Motion is detected in pixel $\mathbf{k}$ if

$$|I(\mathbf{k}, t) - \bar{I}(\mathbf{k})| > \eta \tag{7.12}$$

This method is known as background subtraction method and it is often used in real time surveillance operations.

This method is simple and fast. However it has two major drawbacks. It can not cope with intermittent noise (e.g., moving trees) and it can generate ghost active regions is a background object (e.g., a stopped car) starts to move.

More sophisticated models are often used to cope with the first problem. For example, the pixel intensities are often modeled by mixtures of Gaussians which are able to represent multi modal distributions. This is very important to represent more complex patterns of motion as the ones generated by moving trees. The mixture parameters are often updated during the operation of the system to deal with changes of illumination.

The independent classification of the pixels often leads to small groups of active pixels which can not be interpreted as objects. The binary image produced by the motion detector is often processed by morphological filters to remove small regions and fill the holes. This operation is performed in a suboptimal way since the post-processing criteria are different from the segmentation criterion. This difficulty can be solved by performing a joint classification of the image pixels.

7.5 Joint Pixel Classification

The section addresses the joint classification of all the pixels: *can we enforce labeling coherence to avoid spurious labels?* The answer is yes and it leads to powerful (but time consuming) segmentation techniques. The basic idea is to include some prior information about label distribution.

A simple way to encode this information is by saying that we want a small number of label changes among neighboring pixels. This can be done by defining a prior distribution $P(x;\alpha)$ for the label sequence $l = (l_0, \ldots, l_{n-1})$ (α represents a parameter to be estimated). This can be done using Markov random fields (see Chapter 4) e.g., the Ising model.

Assuming a prior distribution, image segmentation can be performed by computing the most probable labeling i.e., by maximizing

$$P(l|u) = c\, p(u|l)\, P(l;\alpha) \tag{7.13}$$

This approach allows important improvements is most problems. However, the optimization of $P(l|u)$ is usually a difficult combinatorial problem.

The segmentation of the image can also be formulated in a deterministic framework. Let us suppose that there is a classification cost $E_k(l_k)$ when we assign a label l_k to the $k-th$ pixel and let $E_{int}(l)$ be the cost of a labeling configuration x. For example, labeling configurations with a larger number of transitions will have larger costs.

The energy to be minimized has two terms, a data term and a regularization term

$$E(l) = \sum_{k=0}^{n-1} E_k(l_k) + E_{int}(x) \tag{7.14}$$

This approach is equivalent to the the most probable labeling (7.13) if we define

$$E(l) = -\log P(l|u) \tag{7.15}$$

In this case

$$E(l) = -\log P(u|l) - \log P(l|\alpha) \tag{7.16}$$

$$E(l) = -\sum_{k=0}^{n-1} \log P(u_k|l_k) - \log P(l|\alpha) \tag{7.17}$$

Comparing with (7.13) we get an expression for the data energy and for the internal energy

$$E_k(l_k) = -\log P(u_k|l_k) \tag{7.18}$$

$$E_{int}(x) = -\log P(l|\alpha) \tag{7.19}$$

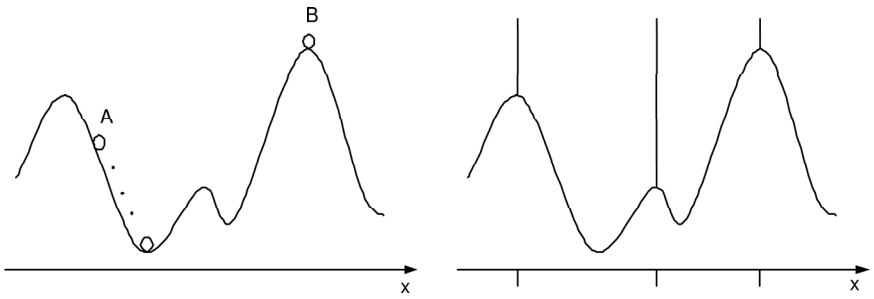


Figure 7.5: Watershed Transform: a) two types of drop locations b) catchment basins and watershed lines

The two approaches (deterministic and probabilistic) are therefore equivalent in this case.

One possible choice for the internal energy is to make it proportional to the number of label changes among all neighboring pixels

$$E_{int}(l) = \alpha \sum_{p,q \in \mathcal{N}} \delta(l_p, l_q) \tag{7.20}$$

where $\mathcal{N}$ is the set of all pairs of neighbors and δ is the Kronecker symbol ($\delta(i,j) = 1$ if $i = j$ and $\delta(i,j) = 0$ otherwise).

The optimization of E is a non-trivial task since it is an combinatorial problem with a very large number of variables. The ICM algorithm and the Gibbs sampler are popular choices for many years. However they have been outperformed by min cut / max flow algorithms which are able to obtain the optimal segmentation in polynomial (quasi-linear) time. In the binary case, these algorithms achieve optimal solutions for the segmentation task with a small computation time and they have already been used in real time surveillance applications.

7.6 Watershed Transform

The watershed transform (WT) is inspired in the ways water appears and spreads in nature. Suppose we interpret the image as a topographic map. The image intensity $I(x)$ stands for the elevation at the point of coordinates x and suppose we place a drop of water on each point of the surface. Most of the drops will slide along the surface until they reach a local minimum (see Fig. 7.5). Few drops will not move since they are located on a local maximum. These drops will converge to more than one minima if they are slightly disturbed.

We denote by *watershed* or *catchment basin* the attraction region of each minimizer x^* of the

Figure 7.6: Watershed segmentation: a) original image, b) gradient intensity, c) watershed segmentation

image I. If we place a drop of water at each point of a catchment basis they will converge to the same minimizer. The points which become attracted to multiple minima if a small disturbance is applied to the drop location belong to *watershed lines*.

The set of all catchment basins defines a segmentation of the image. Each region (object) is associated to a local minimum of the image intensity.

The WT is a useful tool if we want to segment dark objects in a bright background since each object will be represented by a catchment basin. If we have objects with different intensities it is often better to apply the WT to the magnitude of the gradient instead $\nabla u(x)$ since the gradient will be small inside each object. Figure 7.6a shows an image with several homogeneous patches. Figure 7.6b,b display the gradient magnitude and the watershed segmentation.

How can we compute the watershed basins? we could simulate the water drop fall starting from each pixel of the input image. This was the first approach but it is very time consuming.

A more interesting approach is based on immersion i.e., it assumes that the topographic surface is being flooded by water and the water level is increasing [9].

Let us assume that there is a hole located at each local minimum of the topographic surface. When the water level increases, the water starts to flood the deepest basins first and then all the others. Suppose the water in each basin has a different color. We want to build dams in such a way that the water from different basins does not mix and colors are preserved. The dams are built during the flood process. At the end of this process the dams define the watershed lines which define the segmentation of the image. Figure 7.7 illustrates the flood process.

How can we implement dam construction in a computer? one way to do this in by precomputing the set T_p of all pixel locations x such that $I(x) \leq p$. This set defines all the regions

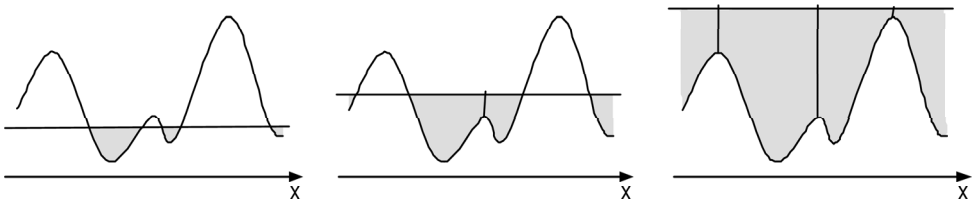


Figure 7.7: Flood process and dam construction

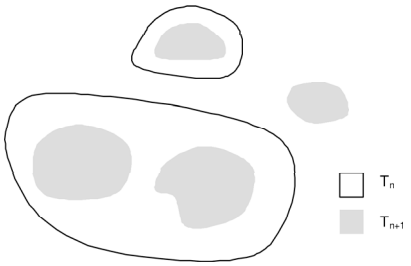


Figure 7.8: Region creation and merging

which were flooded by water at level p. When we change from level p to level $p+1$, two situations must be detected (see Fig. 7.8): i) the presence of new connected components and ii) the merge of connected components (which should be prevented by building dams).

The first situation can be easily detected by computing the connected components of T_{p+1} and checking if they all contain connected components of T_p. If not a new basin is created. The second situation is also easily detected from the connected components of T_n and T_{p+1}. If a connected component of T_{p+1} contained more than one connected component of T_p we have a region merge which should be prevented. The dam can be constructed by performing a set of morphological dilations of the connected components of T_n until they meet. In this process the connected components should not grow outside T_{p+1}.

The constrained dilation of the regions $C, C' \subset T_p$ is described by

$$\begin{aligned} C &\leftarrow T_{p+1} \cap D(C) \\ C' &\leftarrow T_{p+1} \cap D(C') \end{aligned} \tag{7.21}$$

where $D(.)$ stands for the dilation operator.

Figure 7.9: Active contours

7.7 Active Contours

Active contours try a different approach instead of performing region labeling they try to fit the object boundary by a deformable curve. The curve is initialized in the vicinity of the object boundary and it is attracted toward the boundary by forces acting on the curve points (see Fig. 7.9).

The idea is very attractive. The main difficulty is how to attract the curve toward the object boundary (see Figure 7.10). How can this be done since we do not know the boundary points and we do not have a reliable criterion to do the match?

To overcome this difficulty Kass et al. proposed the use of a potential function $P(x)$ computed from the image (see Fig. 7.10(right)). P is chosen in such a way that intensity transitions become associates to valleys of the potential function. The contour model is then estimated by attracting each point toward the valleys.

We will now formalize these ideas. Let $x : [0, 1[\rightarrow \mathbb{R}^2$ be the parametrization of a curve in the image domain [1]. We associate an energy to each curve configuration

$$E(x) = \int_0^1 P(x(s))ds + \int_0^1 \alpha||x'(s)||^2 + \beta||x''(s)||^2 ds \tag{7.22}$$

where $(.)'$ denotes a derivative with respect to s. The energy has two terms: an image dependent term (external energy) and regularization ter (internal energy) which tries to keep the curve derivatives small. This energy measures two conflicting goals. It measures the fitness of the model to the data and tries to keep the shape smooth.

The minimization of $E(x)$ is a problem of variational calculus. We wish to find the function $x(s)$ which minimizes an integral of the form

[1] we assume in this section that the image domain is $\mathbb{R}^2$

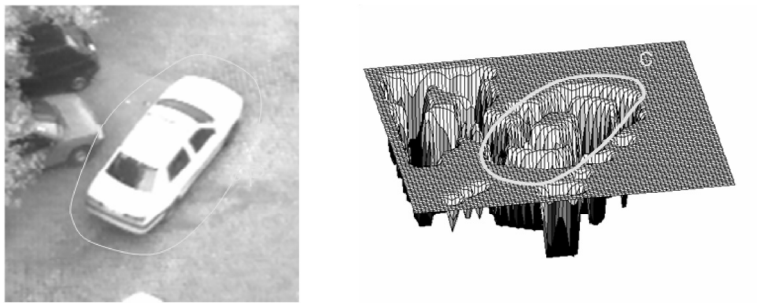


Figure 7.10: Assignment problem: which boundary point should we choose? (left) potential function (right)

$$E = \int_0^1 F(x, x', x'')ds \tag{7.23}$$

A necessary condition for optimality is given by the Euler equation. The Euler equation in this case is given by (see [7] and the proof in Appendix)

$$\alpha x''(s) - \beta x''''(s) - \frac{dP}{dx} = 0 \tag{7.24}$$

This is a nonlinear differential equation whose solution is a stationary point of the energy. If we define the image force at the curve point $x(s)$ as being the gradient of the potential at that point

$$F_{img}(x) = -\frac{dP}{dx} \tag{7.25}$$

and the internal forces as

$$F_{int} = \alpha x''(s) - \beta x''''(s) \tag{7.26}$$

then the Euler equation defines an equilibrium of forces applied to each point of the curve

$$F_{img}(x) + F_{int}(x) = 0 \tag{7.27}$$

This should be true for the optimal contour configuration.

To minimize E we can define a dynamic Euler equation which describes the evolution of the curve deformed by the internal and external forces.

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{F}_{img}(\mathbf{x}) + \mathbf{F}_{int}(\mathbf{x}) \tag{7.28}$$

where $\mathbf{x}$ is a function of s and t. This equation usually converges to a stationary solution $\mathbf{x}(s)$ which verifies the (static) Euler equation as we wished.

To apply this equation we need to define the image potential. Several proposals have been made. Most of them try to make the potential small in the vicinity of the image transitions associated with the object boundary. Some choices are

$$P(\mathbf{x}) = -\frac{d}{d\mathbf{x}}(h(\mathbf{x}) * u(\mathbf{x})) \tag{7.29}$$

where $h(x)$ is the impulse response of a low pass filter or

$$P(\mathbf{x}) = -\sum_{\mathbf{x}_0 \in T} G_\sigma(\mathbf{x} - \mathbf{x}_0) \tag{7.30}$$

where T is the set of all the edge points in the image and

$$G_\sigma(\mathbf{x}) = \frac{1}{2\pi\sigma^2} e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}} \tag{7.31}$$

is a Gaussian Kernel which spreads the influence of each edge point x_0 inside a circle.

Discrete Snake

The previous approach cannot be implemented directly since the curve is continuous and depends on an infinite number of parameters. To circumvent this difficulty the curve can be represented by a B-spline or approximated by a discrete sequence of points (see [6]).Let us consider the second case. The curve $\mathbf{x} : [0, 1[\rightarrow \mathbb{R}^2$ is replaced by a sequence of $2D$ points: $\mathbf{x}(0), \mathbf{x}(1), \ldots, \mathbf{x}(N-1)$.

To estimate the curve configuration the dynamic Euler equation will be discretized. We will assume that t is an integer variable and replace the time derivative in (7.28) by a first order difference

$$\frac{\partial \mathbf{x}(i,t)}{\partial t} \rightarrow \mathbf{x}(i, t+1) - x(i,t) \tag{7.32}$$

Therefore,

$$\mathbf{x}(i, t+1) = \mathbf{x}(i,t) + \mathbf{F}_{img}(\mathbf{x}(i,t)) + \mathbf{F}_{int}(\mathbf{x}(i,t)). \tag{7.33}$$

This equation is valid for all snake points i.e., for $i = 0, \ldots, N-1$. We can write this equation in a compact way using matrix notation.

Let us define two matrices containing all the contour samples and the corresponding image forces,

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{x}(0,t)^T \\ \vdots \\ \mathbf{x}(N-1,t)^T \end{bmatrix} \qquad \mathbf{F}_{img}(t) = \begin{bmatrix} \mathbf{F}_{img}(\mathbf{x}(0,t))^T \\ \vdots \\ \mathbf{F}_{img}(\mathbf{x}(N-1,t))^T \end{bmatrix}. \tag{7.34}$$

$\mathbf{x}(t), \mathbf{F}_{img}$ are $2 \times N$ matrices. Then the snake update can be written as follows

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{M}\mathbf{x}(t) + \mathbf{F}_{img}(\mathbf{x}(t)) \tag{7.35}$$

where $\mathbf{M}$ is a circulant pentadiagonal matrix[2]

$$\mathbf{M} = \begin{bmatrix} a & b & c & 0 & \dots & 0 & c & b \\ b & a & b & c & \dots & 0 & 0 & c \\ c & b & a & b & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & & \vdots & \vdots & \vdots \\ c & 0 & 0 & 0 & \dots & b & a & b \\ b & c & 0 & 0 & \dots & c & b & a \end{bmatrix} \tag{7.36}$$

where a, b, c depend on the regularization parameters α, β as follows: $a = -2\alpha - 6\beta, b = \alpha + 4\beta, c = -\beta$; $\mathbf{M} = \mathbf{0}$ if no regularization forces are considered.

Equation (7.35) provides a simple recursive rule to update the snake contour until it converges to a final configuration. The final configuration depends on the initialization.

Figure 7.11 shows the estimation of the coronary vessels using snakes. These results were obtained using $N = 30$ samples and $\alpha = 1$. Figure 7.11a shows the original contours. Figure 7.11b displays the convergence of the snake algorithm and figure 7.11c displays the final contours when convergence is achieved. The final contour depends on the initialization as shown in this example.

Snakes converges to the object boundary provided they are initialized close enough to the final configuration. They have however several limitations since they are attracted by other objects as well or by the clutter of the image background. They also perform poorly in the presence of concavities. These limitations can be partially overcome by using more sophisticated forces of improved shape models. For example if we have a statistical representation of the object shape derived from examples [8].

Discussion

The snake converges to the object boundary if it is initialized close enough. However it has several limitations since it is also attracted toward other objects or to the background noise (clutter). These limitations can be aliviated by using better image forces and improving the

[2] a matrix is called circulant iif each line is obtained by a circular shift of the previous line.

Figure 7.11: Snake results: a) original image and initial contours; b) potential function and contour configuration at iteration $20, 50, 100, 150, 200$ and c) final contour

shape models. For example, statistical models of the desired shape can be used to make the shape estimate converge towards typical shape configurations[8].

Exercises

- Derive an algorithm to update the discrete snake $\mathbf{x} = (\mathbf{x}(0), \mathbf{x}(1), \ldots, \mathbf{x}(N-1))$ by in order to minimize the energy

$$E = \sum_{i=0}^{N-1} P(\mathbf{x}(i)) + \alpha \sum_{i=0}^{N-1} \|\mathbf{x}(i) - \mathbf{x}(i-1)\|^2 \tag{7.37}$$

  assuming that $\mathbf{x}(-1) = \mathbf{x}(N-1)$. Hint: the derivative of E with respect to $\mathbf{x}(p)$ must be zero.

- Consider the discrete signal $I = (4, 2, 1, 1, 3, 5, 4, 3, 3, 2, 2, 4, 5, 6)$. Compute the watershed transform suing the immersion algorithm.

Appendix - Euler Equation

The Euler equation is derived using variational calculus (e.g., see [7]). We will briefly derive the Euler equation for this case.

Let $\mathbf{x} : [0, 1[\rightarrow \mathbb{R}^2$ be a curve minimizing the energy (7.22). Let us modify $\mathbf{x}$ keeping the boundary conditions invariant. This can be done by adding a perturbation

$$\tilde{\mathbf{x}}(s) = \mathbf{x}(s) + \epsilon\, \eta(s) \tag{7.38}$$

where ϵ is a constant and η is an arbitrary function such that $\eta(0) = \eta(1) = 0, \dot{\eta}(0) = \dot{\eta}(1) = 0$.

If ϵ varies, the energy should have a minimum for $\epsilon = 0$ and

$$\left.\frac{dE}{d\epsilon}\right|_{\epsilon=0} = 0 \tag{7.39}$$

This stationary condition is the Euler equation. Let us now write it in a more explicit way taking the expression of E into account

$$\frac{d}{d\epsilon}\int_0^1 P(\mathbf{x}+\epsilon\eta) + \frac{\alpha}{2}\|\mathbf{x}'+\epsilon\eta'\|^2 + \frac{\beta}{2}\|\mathbf{x}''+\epsilon\eta''\|^2\, ds = 0 \tag{7.40}$$

$$\int_0^1 \frac{d}{d\epsilon}[P(\mathbf{x}+\epsilon\eta) + \frac{\alpha}{2}\|\mathbf{x}'+\epsilon\eta'\|^2 + \frac{\beta}{2}\|\mathbf{x}''+\epsilon\eta''\|^2]\, ds = 0 \tag{7.41}$$

Computing the derivative at $\epsilon = 0$ using the chain rule,

$$\int_0^1 \left(\frac{dP}{d\mathbf{x}}\right)^T \eta + \alpha \underbrace{\mathbf{x}'^T\eta'}_{A} + \beta \underbrace{\mathbf{x}''^T\eta''}_{B}\, ds = 0 \tag{7.42}$$

Integrating by parts terms A and B we obtain

$$\int_0^1 \mathbf{x}'^T\eta'\, ds = \mathbf{x}'^T\eta\big|_0^1 - \int_0^1 \mathbf{x}''^T\eta\, ds \tag{7.43}$$

$$\int_0^1 \mathbf{x}''^T\eta''\, ds = \mathbf{x}''^T\eta'\big|_0^1 - \int_0^1 \mathbf{x}'''^T\eta'\, ds \tag{7.44}$$

$$\int_0^1 \mathbf{x}''^T\eta''\, ds = \mathbf{x}''^T\eta'\big|_0^1 - \mathbf{x}'''^T\eta\big|_0^1 + \int_0^1 \mathbf{x}''''^T\eta\, ds \tag{7.45}$$

Replacing these integrals in (7.42) and using the initial conditions assumed we obtain

$$\int_0^1 [\frac{dP}{d\mathbf{x}} - \alpha\mathbf{x}'' + \beta\mathbf{x}'''']^T\eta\, ds = 0 \tag{7.46}$$

This equation is valid for any function η. Therefore, (see [7])

$$\frac{dP}{d\mathbf{x}} - \alpha\mathbf{x}'' + \beta\mathbf{x}'''' = 0 \tag{7.47}$$

and the this the equilibrium equation used in the snake algorithm.

Bibliography

[1] A.A. Amini, T.E. Weymouth, R.C. Jain, Using dynamic programming for solving variational problems in vision. PAMI 12, 855-867, 1990.

[2] J.C Nascimento, J.S. Marques, Performance evaluation of object detection algorithms for video surveillance, IEEE Transactions on Multimedia, Vol. 8, 761-774, 2006.

[3] L. Vincent, o. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, IEEE Trans. on PAMI, 13(6):583 - 598, 1992.

[4] Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In Proc. First International Conference on Computer Vision, 259-268, June 1987.

[5] L. D. Cohen, I. Cohen, Finite element methods for active contour models and balloons for 2-D and 3-D images, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-15(11), 1993.

[6] A. Blake, M. Isard, Active Contours, Springer, 1998.

[7] R. Courant, F. John, An Introduction to Calculus and Analysis, Wiley, 1974.

[8] T.F. Cootes, D. Cooper, C.J. Taylor and J. Graham, Active Shape Models - Their Training and Application, Computer Vision and Image Understanding. Vol. 61, No. 1, 38-59, 1995.

[9] L. Vincent, o. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, IEEE Trans. on PAMI, 13(6):583 - 598, 1992.

Chapter 8

Object Recognition

8.1 Introduction

Object recognition is a hot topic for more than three decades. Yest, it is not solved. Everyday thousands of images are produced in many contexts. Some of them are available on the net. We must have methods to access this information an efficient ways according to its context i.e., we need a visual google. Other recognition problems are signature recognition, handwritten letter recognition, faced recognition, car plate recognition and building recognition using aerial photographs.

Some recognition tasks are simple. This happens if the object is easily detected in the image and can be easily characterized by its shape, motion or color. We have also extremely efficient algorithms to detect rigid objects with a known texture (e.g., building detais) invariant to the camera point of view and distance. However, it is still difficult to detect and recognize objects whose shape and texture presents a wide variety of different configurations (e.g., animals).Figure ? shows several types of objects with different difficulty levels.

Sometimes we do not want to recognize an object from a wide set classes. We just want to know if an object of a specific type (face, mine) is present in an image and where it is located. This problem is known as object detection.

We can formalize the recognition and detection problems as follows:

Problem 1: object recognition Given an image we and to recognize all the objects despite their position and orientation with respect to the camera.

Figure 8.1: Different object recognition challenges.

A special case which occurs in many applications (e.g., surveillance and military applications) is the detection of objects of interest (persons, planes, boats) in image and video. We can formulate that problem as follows.

Problem 2: object detection Given an image we wish to detect the location of all the objects of a specific classe and their location.

Object recognition problems may have a wide range of difficulties. The recognition of 2D objects with known shape and texture is easy even if the shape and texture information is randomly modified. There are also efficient techniques to deal with 3D objects with known texture observed from arbitrary points of view. Some of these techniques are rather recent [15]. The main challenge concerns 3D object recognition when the shape, color and texture dramatically change as it happens with animals (e.g., horses, dogs) and most objects in our daily life (cars, bycicles, flowers, chairs, tables, etc).

Current systems are not able to solve the general recognition problem but they can provide acceptable solutions for many of the simpler problems. Let us discuss some examples.

Examples

License Plate Recognition

Vehicles are identified by license plates. An automatic recognition of the license plate is a useful for surveillance and toll enforcement in highways. The image of the plate is obtained and processed to compensate for motion blurring and geometric deformation. Then character segmentation is performed and a character recognizer is used to retrieve the license plate number.

The last step is a typical Pattern Recognition problem which can be solved by standard techniques such as support vector machines, classification trees or neural networks.

Detection of Moving objects

It is often useful to detect moving objects (e.g., people, vehicles, boats) in images with cluttered backgrounds. This problem is solved by finding useful features (motion, color, texture) to discriminate the object from the background.

One way to address this problem is by dividing the image into overlapping blocks and make a binary decision based on the block featires (motion, color, texture). This can be done by training a classifier algorithm as in the previous example.

Silhuette Recognition

The silhuette of an object (outer boundary) contains useful information to recognize the object. If the silhuette is a useful feature we can recognize the object using a 2D shape classifier. There are several shape features such as chain codes, invariant moments and Fourier descriptors, which can be used for this purpose.

Face Detection

Face detection is a very important problem in surveillance applications. The goal is to detect all faces present in an image. Current methods consider overlapping blocks and provide a binary decision for each block using a classifier. A sucessful example is [5].

Category Recognition

This problem occurs when was wish to analyse large amounts of images from different sources (e.g., web). We wish to recognize the type of objects included in each image (e.g., vehicles, buildings, sky, animals). The main difficulty concerns the variety of shapes, texture and color within each class as well as the need for efficient methods able to cope huge amounts of information.

8.2 Silluette Analysis

The object silluette (boundary) is a useful cue for recognition. Several methods try to recognize objects from their silluette characterized by a closed curve $\gamma : [0, L[\rightarrow \mathbb{R}^2$ describing the object

boundary or by a binary image in which each point has a binary value $0, 1$ depending on whether it belongs to the object or not.

The same object may appear at different positions in the image and often with different scales and orientations. The recognition method should remain invariant or at least robust with respect to geometric transformations which may occur during the acquisition process.

Region Comparison

The simplest strategy consists of comparing two binary images directly on a pixel by pixel basis.

Let I be a binary image representing an object of interest ($I(\mathbf{x}) = 1$ iif $\mathbf{x}$ to the object) and let M be a binary image representing the object model.

We can compare the images directly and check if there is a strong superposition between the regions marked with 1. Let A, B be the sets of image points with label 1 in the images I, M, respectively. The error between both sets of points can be measured by counting the pixels belonging to only one of the regions, and normalize this number by the object area

$$D = \frac{\#[(A \cup B) - (A \cap B)]}{\#(A \cup B)} \tag{8.1}$$

This method is simple but it has major drawbacks since it assumes that the objects are aligned. It does to deal with translations, rotations or scaling of the objects.

Invariant Moments

A binary object can be characterized by their central moments computed as follows. Let I be a binary image of the object. The center moment of order p, q is given by

$$\mu_{pq} = \sum_{(x_1,x_2)\in\mathbb{Z}^2} (x_1 - \bar{x}_1)^p (x_2 - \bar{x}_2)^q I(x_1, x_2) \tag{8.2}$$

The center moments are invariant with respect to translation and they can be modified in order to enforce invariance with respect to rotation and scaling.

Invariance with respect to scaling can be easily obtained by normalizing the center moments as follows

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(p+q)/2}} \tag{8.3}$$

The invariance with respect to rotation is mutch harder but it can be achieved by the

nonlinear transformation proposed by Hu [1, 2]

$$
\begin{aligned}
I1 &= \eta_{20} + \eta_{02} \\
I2 &= (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \\
I3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \\
&\quad \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
I6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})] \\
I7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].
\end{aligned}
\tag{8.4}
$$

This set of features are known as invariant moments and they have been used for object recognition.

These features have an important drawback. Higher order models are very sensitive to small changes of the boundary since they lead to new terms multiplied by large factors $(x-\bar{x})^p(y-\bar{y})^q$. This effect amplifies small errors and makes the recognition more difficult.

Chain Codes

Uma representação alternativa consiste em dividir o contorno do objecto γ em troços equiespaçados e calcular a direcção de cada troço em relação ao troço anterior. A mudança de direcção é tipicamente quantificada em 8 níveis $\pi i/4$, identificados pelo índice ínteiro $i = 0, \ldots, 7$. A fronteira do objecto é assim caracterizada por uma sequência de inteiros que representam as variações de direcção. Esta representação é conhecida por *código de cadeia diferencial* (*differential chain code*).

O código de cadeia diferencial é invariante em relação a translações e em relação a rotações de múltiplos de $\pi/4$. É ainda aproximadamente invariante em relação a rotações de outras amplitudes.

Uma dificuldade neste método consiste na escolha do ponto inicial a partir do qual se começa a descrever o contorno. Essa escolha é por vezes feita calculando a intersecção do eixo de menor inércia do objecto com a fronteira e tomando um dos dois pontos de intersecção (p.ex., o mais afastado do centro de massa) como ponto inicial.

O reconhecimento de objectos representados por códigos de cadeia passa a ser um problema de comparação de sequências que pode ser abordado por técnicas variadas p.ex., alinhamento

temporal dinâmico, cadeias de Markov não observáveis ou métodos sintáticos de reconhecimento de padrões (gramáticas).

Descritores de Fourier

O objecto pode ser reconhecido através da análise da fronteira. Seja $\gamma(s)$ uma curva fechada descrevendo a fronteira do objecto e $\psi(s)$ uma curva representando um modelo. O parâmetro s define univocamente cada ponto da curva. Por conveniência admitiremos que estamos a trabalhar no plano complexo e $\gamma(s), \psi(s) \in \mathbb{C}$

Há muitas formas differentes de parametrizar uma curva. Há ambiguidade na escolha do ponto inicial que corresponde a $s = 0$ e differentes formas de estabelecer a correspondência entre os pontos da curva e os valores de s. Uma possíbilidade é escolher s de forma a percorrer a curva com velocidade constante e de forma a que cada volta corresponda a um incremento de 2π. As funções $\gamma(s), \psi(s)$ são assim funções periódicas e podem por isso ser decompostas em série de Fourier

$$\gamma(s) = \sum_{k=-\infty}^{\infty} c_k e^{jks} \qquad \psi(s) = \sum_{k=-\infty}^{\infty} d_k e^{jks} \tag{8.5}$$

sendo os coeficientes da série calculados através de produtos internos entre o sinal e as funções de base complexas (exponenciais). A norma L_2 entre as duas funções $\gamma(s), \psi(s)$ pode ser obtida a partir dos coeficientes (relação de Parseval)

$$\|\gamma - \psi\|^2 = \sum_{k=-\infty}^{\infty} |c_k - d_k|^2 \tag{8.6}$$

Contudo esta é uma má medida para o reconhecimento pois requer que os objectos estejam alinhados.

Vale a pena perceber qual a influência das transformações geométricas sobre os coeficientes de Fourier. A tabela 8.1 resume este efeito para cada uma das transformações mencionadas. A translação só altera o coeficiente de ordem 0 (DC). A rotação altera a fase dos coeficientes somando-lhes um termo aditivo linear. As amplitudes ficam invariantes. A mudança de ponto inicial também só altera a fase. A mudança de escala altera as amplitudes mas de uma forma fácil de compensar através de uma normalização.

Um conjunto de descritores invariantes a estas transformações podem ser obtidos usando o módulo dos c_k normalizado

$$\rho_k = \frac{|c_k|}{\sum_{i=1}^{N} |c_i|}, \qquad k > 0 \tag{8.7}$$

em que N é o número de coeficientes usados para normalização. O coeficiente DC foi eliminado e os coeficientes de índice negativo também devido à simetria.

Translação	$\gamma(s) + d$	$c_k + d\delta(k)$
Rotação	$\exp(j\theta)\gamma(s)$	$e^{j\theta}c_k$
Ponto inicial	$\gamma(s - s_0)$	$e^{js_0k}c_k$
Escalamento	$\alpha\gamma(s)$	αc_k

(8.8)

Table 8.1: Coeficientes de Fourier após transformações geométricas

8.3 Ajuste de template

Um método simples para reconhecimento/detecção de objectos consiste em obter uma imagem do objecto de interesse (template) $\mathbf{T}$ e compará-la com blocos de iguais dimensões da imagem observada I. A comparação é feita com base na norma l_2. Assim, calcula-se

$$E(\mathbf{x}_0) = \sum_x (I(\mathbf{x} - \mathbf{x}_0) - T(\mathbf{x}))^2 \tag{8.9}$$

em que $\mathbf{x_0} \in \mathbb{Z}^2$ é o deslocamento aplicado à template. O objecto é detectado numa posição $\mathbf{x_0}$ se E tiver um mínimo local nesse ponto e o mínimo for inferior a um limiar pré-definido

$$E(\mathbf{x_0}) < \lambda \tag{8.10}$$

Este método designa-se por *método de ajuste de template*. A energia E é calculada tipicamente de forma esparsa ao longo de uma grelha de valores de $\mathbf{x_0}$ que corresponde a ir deslocando a janela na imagem. Esta operação deve ser feita com sobreposição entre posições consecutivas da janela e é por isso pesada do ponto de vista computacional.

O método de ajuste de template pode ser rescrito usando notação matricial. Designando por $\mathbf{t}, \mathbf{x}$ vectores obtidos por concatenação das colunas de $\mathbf{T}$ e do bloco da imagem $\mathbf{B}$ vem

$$E = ||\mathbf{x} - \mathbf{t}||^2 \tag{8.11}$$

O deslocamento $\mathbf{x_0}$ está implícito na escolha do bloco $\mathbf{B}$.

8.4 Detecção Probabilística

Seja $\mathbf{B}$ um bloco de imagem que pode conter ou não um objecto extraído de uma colecção de objectos pré-definidos $O_i, i = 1, \ldots, M$ e seja $\mathbf{x}$ um vector contendo todos os pixels de $\mathbf{B}$. O vector $\mathbf{x}$ tem uma dimensão muito elevada (da ordem dos milhares).

O reconhecimento de O_i pode ser formulado como um problema de decisão. Dadas as distribuições do vector de observações para cada uma das classes (objectos) $p(\mathbf{x}|O_i)$, classifica-se $\mathbf{x}$ na classe mais provável

$$P(O_i|\mathbf{x}) = kp(\mathbf{x}|O_i)P(O_i) \qquad i = 0, \ldots, M \tag{8.12}$$

em que $P(O_i|\mathbf{x})$ é a probabilidade *a posteriori* da classe O_i, $P(O_i)$ a probabilidade *a priori* e $k = 1/p(\mathbf{x})$. Adicionou-se uma classe adicional O_0 que designa a hipótese nula (não há objecto em $\mathbf{B}$).

Um caso particular é o do problema de detecção em que há apenas duas hipótese: O ou $\bar{O}$ (presença ou ausência do objecto de interesse em $\mathbf{B}$). Neste caso pretende-se apenas saber se (detecção de objecto)

$$P(O|\mathbf{x}) > P(\bar{O}|\mathbf{x}) \tag{8.13}$$

$$P(\mathbf{x}|O)P(O) > P(\mathbf{x}|\bar{O})P(\bar{O}) \tag{8.14}$$

Por vezes conhecemos apenas a distribuição de $\mathbf{x}$ quando há objecto no bloco $\mathbf{B}$. Neste caso, o teste pode ser aproximado por

$$p(\mathbf{x}|O) > \lambda \tag{8.15}$$

em que $\lambda = p(\mathbf{x}|\bar{O})P(\bar{O})|P(O)$.

Se $\mathbf{x}$ tiver uma distribuição gaussiana quando o objecto está presente, $\mathbf{x} \sim N(\bar{x}, R)$, então a detecção consiste em verificar se a desigualdade é verdadeira

$$Ce^{-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^T R^{-1}(\mathbf{x}-\bar{\mathbf{x}})} > \lambda \tag{8.16}$$

ou seja

$$d(\mathbf{x}) < \alpha \tag{8.17}$$

$$d(x, \mathbf{x}) = (\mathbf{x} - \bar{\mathbf{x}})^T R^{-1}(\mathbf{x} - \bar{\mathbf{x}}) \tag{8.18}$$

é a distância de Mahalanobis e $\alpha = -2\log(\lambda/C)$.

O vector de média e a matriz de covariância podem ser estimados a partir de um conjunto de treino de blocos $\{\mathbf{x}^i, i = 1, \ldots, N\}$ por

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^i \tag{8.19}$$

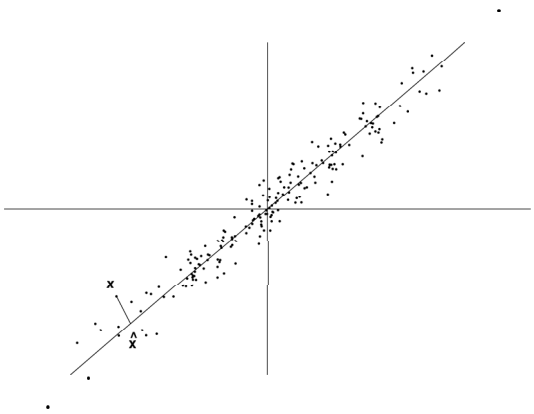


Figure 8.2: Estimação de subespaço

$$\mathbf{R} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^i - \bar{\mathbf{x}})(\mathbf{x}^i - \bar{\mathbf{x}})^T \tag{8.20}$$

Infelizmente as coisas não são tão simples. A dimensão muito elevada de $\mathbf{x}$ ($\sim 10^4$) torna impraticável o cálculo da matriz de covariância, nem é possível reunir imagens de treino suficientes ($>> 10^8$) de forma a garantir uma boa estimativa de $\mathbf{R}$. Estas dificuldades são tratadas na secção seguinte.

8.5 Reconhecimento com PCA

A análise em componentes principais é usada para reduzir a dimensão dos dados.

Consideremos o caso mais simples em que se pretende aproximar um vector aleatório $\mathbf{x} \in \mathbb{R}^n$, de média nula, por um vector pertencente a um subespaço com um único vector de base, $\hat{\mathbf{x}} = \alpha\phi, \phi \in \mathbb{R}^n$ (ver figura 8.2). Admitiremos que o vector ϕ está normalizado ou seja $\|\phi\| = 1$. O erro de aproximação é $\mathbf{e} = \mathbf{x} - \alpha\phi$.

Pretende-se estimar α, ϕ de forma a que o erro quadrático médio

$$E = E\{\mathbf{e}^T\mathbf{e}\} = E\{\mathbf{x}^T\mathbf{x} - 2\alpha\phi^T\mathbf{x} + \alpha^2\phi^T\phi\} \tag{8.21}$$

seja mínimo. O valor óptimo para o coeficiente é $\alpha = \mathbf{x}^T\phi$ e como $\phi^T\phi = 1$ vem

$$E = E\{\mathbf{x}^T\mathbf{x} - \phi^T\mathbf{x}\mathbf{x}^T\phi\} = tr(\mathbf{R}) - \phi^T\mathbf{R}\phi \tag{8.22}$$

em que $\mathbf{R} = E\{\mathbf{x}\mathbf{x}^T\}$ é a matriz de covariância de $\mathbf{x}$. A minimização de (8.22) sujeita à restrição de $\phi^T\phi = 1$ obtém-se minimizando a função lagrangiana

$$L = tr(\mathbf{R}) - \phi^T\mathbf{R}\phi + \lambda(\phi^T\phi - 1) \tag{8.23}$$

Calculando a derivada de L em ordem a ϕ e igualando a zero obtém-se

$$\mathbf{R}\phi = \lambda\phi \tag{8.24}$$

que é uma equação verificada pelos valores e vectores próprios da matriz $\mathbf{R}$. Assim, ϕ deve ser um vector próprio da matriz de covariância $\mathbf{R}$. Resta saber qual dos valores próprios devemos escolher para obter um mínimo. Substituindo ϕ em (8.22) obtém-se

$$E = tr(\mathbf{R}) - \lambda \tag{8.25}$$

Assim, a energia E é minimizada se ϕ for o vector próprio associado ao maior valor próprio de $\mathbf{R}$. Este vector corresponde à direcção de maior variação de $\mathbf{x}$ e é também conhecido por eixo de menor inércia.

Consideremos agora o caso geral. Se quisermos aproximar $\mathbf{x}$ por um vector pertencente a subespaço de dimensão $d < n$ o procedimento é idêntico ao anterior. O subespaço óptimo é o subespaço gerado por d vectores próprios da matriz $\mathbf{R}$ associados aos maiores valores próprios. Assim

$$\hat{\mathbf{x}} = \sum_{i=1}^{d} y_i\phi_i \tag{8.26}$$

com $y_i = \mathbf{x}^T\phi_i$. Os vectores ϕ_i representam os principais modos de variação de $\mathbf{x}$ e são conhecidos por *imagens próprias* [4]. A equação (8.26) traduz a decomposição de $\mathbf{x}$ como uma combinação de imagens próprias.

É fácil provar que o erro de aproximação é uma combinação linear dos restantes vectores próprios

$$\mathbf{e} = \sum_{i=d+1}^{n} y_i\phi_i \tag{8.27}$$

e a energia do erro é a soma dos $n - d$ valores próprios mais pequenos ou seja,

$$E = \sum_{i=d+1}^{n} \lambda_i \tag{8.28}$$

admitindo que os valores próprios de $\mathbf{R}$ se encontram ordenados por ordem decrescente.

Os coeficientes y_i designam-se por componentes principais de $\mathbf{x}$ e a representação de $\mathbf{x}$ como combinação linear dos vectores próprios da matrix de covariância designa-se por análise em componentes principais.

A análise em componentes principais pode ser expressa em notação matricial. Seja $\mathbf{\Phi} \in \mathbb{R}^{n\times n}$

$$\mathbf{\Phi} = [\phi_1 \dots \phi_n] \tag{8.29}$$

a matriz de vectores próprios de $\mathbf{R}$. Como os vectores próprios de uma matriz simétrica são ortogonais (e foram normalizados), a matriz $\mathbf{\Phi}$ é unitária ou seja,

$$\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I} \tag{8.30}$$

e $\mathbf{\Phi}^{-1} = \mathbf{\Phi}^T$. Por isso, a transformação entre a imagem e os coeficientes pode ser escrita na forma

$$\mathbf{x} = \mathbf{\Phi}\mathbf{y} \qquad\qquad \mathbf{y} = \mathbf{\Phi}^T\mathbf{x} \tag{8.31}$$

Esta transformação é conhecida por transformação de Karhunen-Loeve ou por análise em componentes principais (PCA) [3].

A transformada de Kahunen-Loeve goza das seguintes propriedades.

Transformação unitária:

A matriz de transformação $\mathbf{\Phi}$ é uma matriz unitária $\mathbf{\Phi}^{-1} = \mathbf{\Phi}^T$. Por isso a transformação diz-se unitária.

Invariância da norma:

A transformação $\mathbf{\Phi}$ preserva o comprimento dos vectores. Se $\mathbf{y} = \mathbf{\Phi}\mathbf{x}$, então $\|\mathbf{y}\| = \|\mathbf{x}\|, \forall\mathbf{x}$ ($\|.\|$ designa a norma euclidiana).

Diagonalização de R:

A matriz de covariância verifica a propriedade

$$\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi} = \Lambda \tag{8.32}$$

em que Λ é a matriz diagonal definida pelos valores próprios da matriz $\mathbf{R}$

$$\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_n) \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \tag{8.33}$$

Estimação de subespaço:

Pretende-se aproximar os vectores $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ com covariância $\mathbf{R}$, por vectores (projecções) $\hat{\mathbf{x}}$ pertencentes a um subespaço $S \subset \mathbb{R}^n$ de dimensão $d < n$. O subespaço que minimiza o erro quadrático médio

$$E = E\{\|\mathbf{x} - \hat{\mathbf{x}}\|^2\} \tag{8.34}$$

é o subespaço gerado gerado pelos vectores próprios associados aos m maiores valores próprios de $\mathbf{R}$ ou seja $\mathbf{\Psi} = [\mathbf{\Phi}_1 \ldots \mathbf{\Phi}_d]$ e

$$\hat{\mathbf{x}} = \Psi\hat{\mathbf{y}} \qquad\qquad \hat{\mathbf{y}} = \Psi^T\mathbf{x} \qquad\qquad \hat{\mathbf{y}} \in \mathbb{R}^d \tag{8.35}$$

De entre todas as transformações $\Psi = [\Phi_1 \dots \Phi_d]$ é a transformação óptima no sentido de minimizar o erro quadrático médio de reconstrução. Esta é uma importante propriedade da análise em componentes principais. As componentes principais são as melhores variáveis para reconstruir o sinal a partir de informação incompleta.

O sinal x é decomposto em duas componentes ortogonais: sinal de aproximação e erro

$$\mathbf{x} = \hat{\mathbf{x}} + \mathbf{e} = \sum_{i=1}^{d} y_i \Phi_i + \sum_{i=d+1}^{n} y_i \Phi_i \tag{8.36}$$

Como as duas componentes são ortogonais, a energia de $\mathbf{x}$ é separada em duas partes (prove!)

$$E\{\|x\|^2\} = E\{\|\hat{x}\|^2\} + E\{\|e\|^2\} = \sum_{i=1}^{d} \lambda_i + \sum_{i=d+1}^{n} \lambda_i \tag{8.37}$$

A energia de $\mathbf{x}$ é a soma dos primeiros m valores próprios e a energia do erro é a soma dos restantes

$$E\{\|\mathbf{x} - \hat{\mathbf{x}}\|\} = \sum_{i=d+1}^{n} \lambda_i \tag{8.38}$$

O que fizemos até ao momento foi decompor o espaço $\mathbb{R}^n$ em dois subespaços ortogonais. Estes subespaços são muitas vezes interpretados como um subespaço de sinal e um subespaço de ruído. O primeiro é gerado pelos d primeiros vectores próprios de $\mathbf{R}$ e o segundo pelos restantes vectores próprios. A dimensão d do espaço de sinal deve ser calculada de forma a que a energia do erro seja pequena (inferior a um limiar pré-fixado).

Como veremos é fácil estimar bem o subespaço de sinal mas é difícil estimar o subespaço de ruído. Isso tem consequências como veremos.

Vimos na secção anterior que a detecção de objectos é feita com base na distância de Mahalanobis

$$d(\mathbf{x}) = \mathbf{x}^T R^{-1} \mathbf{x} = \mathbf{x}^T \Phi \Lambda^{-1} \Phi^T \mathbf{x} = \mathbf{y}^T \Lambda^{-1} \mathbf{y} \tag{8.39}$$

Assim,

$$d(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{\lambda_i} y_i^2 \tag{8.40}$$

A distância de Mahalanobis também pode ser decomposta em duas partes correspondentes aos subespaços de sinal e de ruído

$$d(\mathbf{x}) = \sum_{i=1}^{m} \frac{1}{\lambda_i} y_i^2 + \sum_{i=m+1}^{n} \frac{1}{\lambda_i} y_i^2 \tag{8.41}$$

Em geral é fácil calcular os valores próprios e os coeficientes y_i associados ao espaço de sinal (1^o termo) mas não ao espaço de ruído (2^o termo).

Para ultrapassar esta dificuldade propõe-se em [4] que a distância de Mahalanobis seja aproximada pela expressão seguinte em que o segundo termo é um valor aproximado do correcto

$$d(\mathbf{x}) = \sum_{i=1}^{d} \frac{1}{\lambda_i} y_i^2 + \frac{1}{\bar{\lambda}} \sum_{i=d+1}^{n} y_i^2 \tag{8.42}$$

em que $\bar{\lambda}$ é a média aritmética dos últimos $n - m - 1$ valores próprios. A energia do erro pode ser facilmente obtida sem calcular os coeficientes y_i pois $\|x - \hat{x}\|^2 = \|x\|^2 - \|\hat{x}\|^2$. Assim,

$$d(\mathbf{x}) = \sum_{i=1}^{d} \frac{1}{\lambda_i} y_i^2 + \frac{1}{\bar{\lambda}} (\|x\|^2 - \|\hat{x}\|^2) \tag{8.43}$$

Este critério de detecção foi extensivamente testado em operações de detecção de caras na base de dados FERET e comparado em concurso com as melhores técnicas existentes na altura tendo obtido um dos melhores resultados [4].

Aspectos computacionais

Não é fácil estimar $\mathbf{R}$ directamente a partir de uma sequência de vectores $\mathbf{x}^i$ devido à elevada dimensão dos vectores ($\sim 10^3$). O número de vectores de treino é geralmente da ordem das dezenas ou das centenas o que significa que a estimativa de $\mathbf{R}$ tem característica nula (a maioria dos valores próprios são nulos) e não é possível realizar uma análise de valores próprios com uma matriz de dimensões tão grandes. Estas dificuldades são facilmente ultrapassadas uma vez que se pretende calcular apenas os primeiros m valores e vectores próprios com $m << n$.

Seja $\mathbf{X} = [\mathbf{x}^1 \dots \mathbf{x}^N]$, $N < n$ uma matriz cujas colunas são os vectores de treino. Os primeiros N valores e vectores próprios de $\mathbf{R}$ podem ser calculados a partir de uma matriz auxilizar de dimensão baixa $N \times n$: $\frac{1}{N}\mathbf{X}^T\mathbf{X}$.

Proposição: [cálculo dos valores e vectores próprios]

Sejam $(\lambda_i, \mathbf{v}_i), i = 1, \dots, N$, a sequência de valores e vectores próprios da matriz $\frac{1}{N}\mathbf{X}^T\mathbf{X}$. Então $(\lambda_i, \mathbf{X}\mathbf{v}_i), i = 1, \dots, N$ são valores e vectores próprios de $\mathbf{R}$.

A demonstração é simples. Baseia-se na decomposição em valores singulares da matriz de dados $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ em que $\mathbf{U}, \mathbf{V}$ são matrizes ortogonais e $\mathbf{D}$ uma matriz diagonal cujos elementos da diagonal s_i são os valores simgulares da matriz R. A matriz de covariância exprime-se facilmente em termos destas matrizes

$$\mathbf{R} = \frac{1}{N}\mathbf{X}\mathbf{X}^T = \frac{1}{N}\mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{V}\mathbf{D}\mathbf{U}^T = \frac{1}{N}\mathbf{U}\mathbf{D}^\mathbf{2}\mathbf{U}^T \tag{8.44}$$

Comparando com a decomposição em valores e vectores próprios da matriz (8.32) conclui-se que os valores próprios de $\mathbf{R}$ são $\lambda_i = s_i^2$ e os vectores próprios são as colunas de U.

Repetindo o mesmo raciocínio com a matriz auxiliar

$$\frac{1}{N}\mathbf{X}^T\mathbf{X} = \frac{1}{N}\mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{U}\mathbf{D}\mathbf{V}^T = \frac{1}{N}\mathbf{V}\mathbf{D}^\mathbf{2}\mathbf{V}^T \tag{8.45}$$

conclui-se que a matriz auxiliar (de baixa dimensão) tem os mesmos valores singulares e portanto os mesmos valores próprios. Os vectores póprios da matriz auxiliar $\mathbf{v}_i$ são differentes no entanto podem ser relacionados com os de .

Se $\lambda_i, \mathbf{v}_i$ forem um par de valor e vector próprio da matriz auxiliar

$$\mathbf{X}^T\mathbf{X}\ \mathbf{v}_i = \lambda_i\mathbf{v}_i \tag{8.46}$$

Multiplicando os dois membros por $\mathbf{X}$ vem

$$\mathbf{X}\mathbf{X}^T(\mathbf{X}\mathbf{v}_i) = \lambda_i(\mathbf{X}\mathbf{v}_i) \tag{8.47}$$

$$\mathbf{R}(\mathbf{X}\mathbf{v}_i) = \lambda_i(\mathbf{X}\mathbf{v}_i) \tag{8.48}$$

Ou seja $\mathbf{X}\mathbf{v}_i$ é vector próprio de $\mathbf{R}$, como queríamos mostrar.

8.6 Modelos de constelação

Os métodos anteriores são aplicáveis quando os objectos são fáceis de segmentar ou têm uma aparência visual com pouca variabilidade. Não permitem resolver problemas gerais de reconhecimento de objectos p.ex., reconhecer vacas, ou cadeiras ou árvores em posições gerais. Uma vaca pode ter várias cores possíveis. Pode estar em várias posições e ter por isso uma forma muito variada. Além disso pode não ser fácil de segmentar.

O problema de reconhecimento genérico de objectos ou de classes de objectos tem um grande interesse é ainda um tema em aberto mas foram dados passos importantes nos últimos 5 anos com vista à sua resolução.

A tendência é a de desenvolver modelos locais para partes do objecto (chifre da vaca, barbatana do tubarão, olho de pessoa) ligados através de um modelo global que descreve a geometria relativa das várias partes (ver Figura 8.3). Esse modelo é conhecido por modelo de constelação e tem sido desenvolvido por Perona e outros [6, 7].

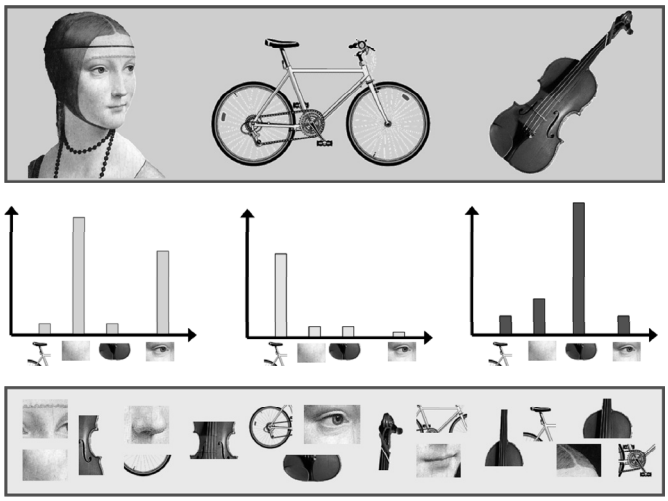


Figure 8.3: Modelos de partes (extraído de [8])

Bibliography

[1] M-K. Hu, Visual pattern recognition by moment invariants, IRE Trans. on Information Theory, 179-187, 1962.

[2] http://homepages.inf.ed.ac.uk/rbf/CVonline

[3] Duda, Hart and Stork, Pattern Classification, Second Edition, Wiley, 2001.

[4] B. Moghaddam, A. Pentland, Probabilistic Visual Learning for Object Representation, IEEE Transactions Pattern Analysis and Machine Intelligence, Vol. 19, 696-710, Julho 1997.

[5] P. Viola, M. Jones, Robust Real-time Object Detection, 2nd International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing and Sampling, 2001.

[6] R. Fergus, P. Perona, A. Zisserman, Object class recognition by unsupervised scale-invariant learning, International Conference on Computer Vision and Pattern Recognition, vol. 2, 264-271, 2003.

[7] M. Weber, Unsupervised Learning of Models for Object Recognition, tese de doutoramento, California Institute of Technology, 2000.

[8] Li Fei-Fei, Rob Fergus, Antonio Torralba, Recognizing and Learning Object Categories, ICCV 2005 short courses, Awarded the Best Short Course Prize, (http://people.csail.mit.edu/torralba/iccv2005/), 2005

Part III

Vision

Chapter 9

Camera Model

9.1 Introduction

The principles of perspective were discovered during the Italian Renaissance by Brunelleschi, da Vinci and other architects and painters. They made drawings and paintings with a remarkable accuracy levels. They found how we see the 3D world. A famous work is Brunelleschi drawing of the church of Santo Spririto in 1436 (see Figure 9.1).

Cameras project 3D points into the image plane. This transformation is not invertible unless we have additional information about the point location in space (e.g., depth). However, it allows us to obtain mutch information about a scene. Using a single camera it is possible to measure the length of a line on the floor, to determine the velocity of a car in a highway, to locate a football player on the field or to validate a goal line[1]. This section is inspired in two excellent textbooks [1, 2].

This chapter shows that camera performs a projection of 3-points in space onto the image plane described by a perspective projection

$$\mathbf{X} = \mathbf{\Pi x} \tag{9.1}$$

where $\mathbf{X}, \mathbf{x}$ are the homogeneous coordinates of the points in space and in the image plane and $\mathbf{\Pi}$ is a 3×4 matrix denoted as camera matrix. The camera matrix can be estimated from the data and fully characterizes the geometric transformation performed by the camera.

[1] recently, researchers from the university of Oxford showed that the third goal of England in the final of World Cup 1966 was not valid: the ball did not cross the line [3].

Figure 9.1: Italian renaissance: Brunelleschi drawing (1436) and photo of the church of Santo Spirito

9.2 Perspective Projection

Digital cameras concentrate electromagnetic radiation (light rays) on the surface of the CCD sensor, using optical lenses. This process can be approximated by the pin hole model which assumes that the lenses are reduced to a small hole (optical center O) and all the light rays arriving at the sensor pass through that hole (see 9.2). Each point P in 3D space is projected into a point p obtained by intersecting the PO straight line with the image plane.

The optical axis of the camera is a straight line orthogonal to the image plane which contains the optical center. The intersection of this line with the image plane is called the principal point of the image.

Let us now consider the frames shown in figure 9.2: the XYZ frame centered at the optical center of the camera, and the image frame xy centered at the principal point of the camera. Let (X, Y, Z) be the coordinates of the point P in the camera frame and let (x, y) be the coordinates of the projected point p on the image plane. It can be easily shown by Tales theorem that

$$x = -f\frac{X}{Z} \qquad y = -f\frac{Y}{Z} \tag{9.2}$$

where f is the focal length of the camera. All these variables are measured in metric coordinates (m). This transformation which maps 3D points into image point is called the *perspective projection.*

The images of the objects obtained using the pin hole model are inverted. To overcome this difficulty we can invert the image. We can do both transformation in a single step by assuming that the image plane is located in front of the optical center O (see Figure 9.3). This is called

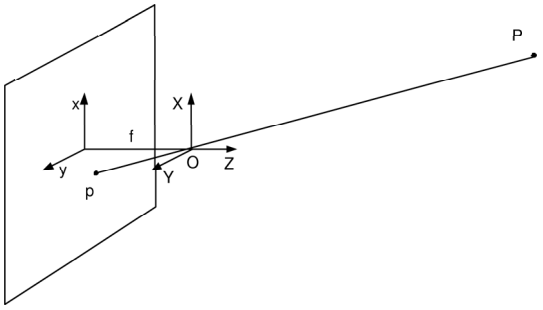


Figure 9.2: Perspective Projection

the frontal model. In this case, the coordinates of the projected point are given by

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z} \tag{9.3}$$

This model is sometimes simplified by assuming $f = 1$ and in this case we call it *normalized model*

$$x = \frac{X}{Z} \qquad y = \frac{Y}{Z} \tag{9.4}$$

The perspective projection is a non linear transformation from the 3D space onto the image plane since the X, Y coordinates are divided by the depth. These equations become simpler if we represent the two points P, p using homogeneous coordinates.

The homogeneous coordinates of two points P, p are defined by

$$\mathbf{X} = \alpha \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \mathbf{x} = \beta \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{9.5}$$

where α, β are arbitrary constants. The homogeneous coordinates are obtained multiplying the Cartesian coordinates by a scale factor and adding the scale factor as an additional coordinate. The scale factor is arbitrary chosen. Therefore, two vectors of homogeneous coordinates $\mathbf{X}, \mathbf{X}'$ represent the same point iif there is a scalar $\lambda \neq 0$ such that $\mathbf{X} = \lambda\mathbf{X}'$. In this section we will use the same notation to denote a point written in Cartesian and in homogeneous coordinates. The meaning should be clear from the context.

The normalized camera model in homogeneous coordinates is given by (check these equa-

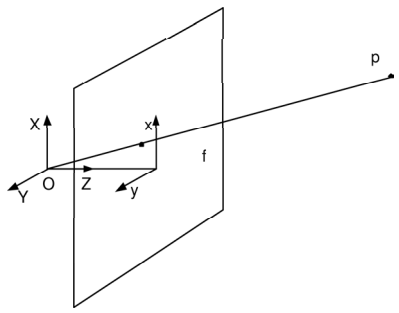


Figure 9.3: Perspective projection (front)

tions)

$$Z\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{9.6}$$

If we define a normalized projection matrix by

$$\mathbf{\Pi_0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = [\mathbf{I}\ \mathbf{0}] \in \mathbb{R}^{3\times 4} \tag{9.7}$$

the camera model is given by

$$\lambda \mathbf{x} = \mathbf{\Pi_0 X} \tag{9.8}$$

where λ is a multiplicative constant (depth).

9.3 Extrinsic Parameters

Until now, we have chosen a 3D frame attached to the camera and located at the optical center of the camera (Figure 9.4). This hypothesis is not appropriate in practice. We can not easily measure the coordinates of point s with respect to the optical center hidden inside the camera.

If we choose another frame with an arbitrary location and orientation, the new coordinates $\mathbf{X_0}$, are related to the camera coordinates by a rigid body transformation (rotation+translation) given by[2]

$$\mathbf{X} = \mathbf{R}\mathbf{X_0} + \mathbf{T} \tag{9.9}$$

[2]we are using Cartesian coordinates in this equation

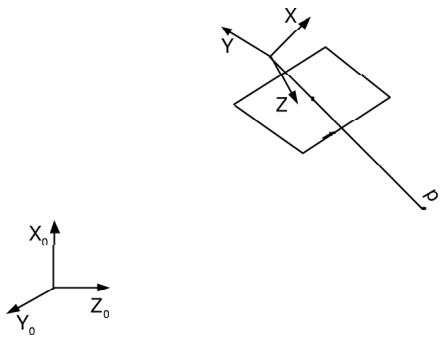


Figure 9.4: Extrinsic parameters

where $\mathbf{X_0} \in \mathbb{R}^3$ is the vector of coordinated of the point measured in the world frame, $\mathbf{R}$) is a rotation matrix, $\mathbf{T} \in \mathbb{R}^3$ is a vector of translation. The parameters of the transformation $(\mathbf{R}, \mathbf{T})$ are called the *extrinsic parameters* of the camera since they depend on the camera location and orientation with respect to the world frame. The rotation matrix belongs to the special group of orthogonal matrices $SO(3)$ i.e., it is an orthogonal matrix ($\mathbf{R^T R} = \mathbf{I}$) and it preserves the orientation of the axis ($\det(\mathbf{R}) = 1$).

The rigid body transformation can easily be written in homogeneous coordinates as follows

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{9.10}$$

That is,

$$\mathbf{X} = \mathbf{g}\mathbf{X_0} \qquad\qquad \mathbf{g} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{9.11}$$

The camera model with extrinsic parameters is given by

$$\lambda \mathbf{x} = \mathbf{\Pi_0 g X_0} \qquad\qquad \lambda \mathbf{x} = [\mathbf{R}\ \mathbf{T}]\mathbf{X_0} \tag{9.12}$$

Example

Consider a camera located in a rectangular room at the height h as shown in the figure. The optical axis of the camera belongs to the plane orthogonal to the wall and it make an angle

of -45^o with respect to the horizontal plane. Given the world frame defined in the figure, we wish to compute the extrinsic parameters of the camera.

Let us now solve this problem. Point p can be represented with respect to two frames: the room frame and the camera frame. These two sets of coordinates are related by $\mathbf{X} = \mathbf{R}\mathbf{X_0} + \mathbf{T}$ where $(\mathbf{R}, \mathbf{T})$ are the parameters of a rigid body transformation.

The versors of the room frame can be expressed in the camera frame as follows

$$\hat{e}_{X_0} = \hat{e}_X \qquad \hat{e}_{Y_0} = \cos\theta\hat{e}_Y - \sin\theta\hat{e}_Z \qquad \hat{e}_{Z_0} = \sin\theta\hat{e}_Y - \cos\theta\hat{e}_Z \tag{9.13}$$

Therefore,

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{9.14}$$

The translation vector should be chosen in such a way that the camera optical center has coordinates $X_0 = [0, 0, h]^T$. Since $\mathbf{R}\mathbf{X}_0 + \mathbf{T} = \mathbf{0}$, we obtain $\mathbf{T} = -h\,[0\ \sin\theta\ -\cos\theta]^T$.

9.4 Intrinsic Parameters

The metric coordinates (m) used so far to specify the location of point on the sensor surface are not practical. Image points are usually measured in pixels and frame is not centered at principal point of the camera.

Therefore, the measured coordinates (x', y') (pixels) are related to (x, y) (m) by

$$x' = fs_x x + o_x \qquad y' = fs_y y + o_y \tag{9.15}$$

where f is the focal length, s_x, s_y are the conversion factors from meters to pixels and o_x, o_y are the coordinates of the principal point in the new frame[3].

This transformation can be expressed in homogeneous coordinates as follows

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & 0 & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{9.16}$$

[3]the focal length f is included in the transformation for the sake of convenience.

In practice we consider a more general transformation assuming that the entry 12 of the matrix may be different from zero. The transformation matrix becomes an arbitrary upper triangular matrix

$$\mathbf{x}' = \mathbf{K}\mathbf{x} \qquad \mathbf{K} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{9.17}$$

The new parameter s_θ has also a geometric meaning: it is a diagonal scale (skew) factor.

The parameters $(fs_x, fs_y, fs_\theta, o_x, o_y)$ are denoted as *intrinsic parameters* of the camera since they only depend of the camera and are independent of the camera position and orientation. Matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ is called the matrix of intrinsic parameters[4].

The camera model with intrinsic and extrinsic parameters is given by

$$\lambda\mathbf{x}' = \mathbf{K}\mathbf{\Pi_0}\mathbf{g}\mathbf{X_0} \tag{9.18}$$

$$\lambda\mathbf{x}' = \mathbf{K}[\mathbf{R}\ \mathbf{T}]\mathbf{X_0} \tag{9.19}$$

The camera model can also be expressed as

$$\lambda\mathbf{x}' = \mathbf{\Pi}\mathbf{X_0} \tag{9.20}$$

where $\mathbf{\Pi} \in \mathbb{R}^{3\times4}$ is a 3×4 matrix denoted as the *camera matrix*. For the sake of simplicity we will replace $\mathbf{X_0}$ by $\mathbf{X}$ from now on.

9.5 Camera Model in Cartesian Coordinates

Let us consider the camera model (9.20). The camera matrix $\mathbf{\Pi}$ can be written as

$$\mathbf{\Pi} = \begin{bmatrix} \pi_1^T \\ \pi_2^T \\ \pi_3^T \end{bmatrix} \tag{9.21}$$

where π_i^T is the i-th line of $\mathbf{\Pi}$. Using (9.20) we obtain $\lambda = \pi_3^T X_0$ and

$$x' = \frac{\pi_1^T \mathbf{X_0}}{\pi_3^T \mathbf{X_0}} \qquad y' = \frac{\pi_2^T \mathbf{X_0}}{\pi_3^T \mathbf{X_0}} \tag{9.22}$$

this is a nonlinear transformation from the 3D space to the image plane known as the *perspective projection*.

[4] a zoom operation changes matrix K.

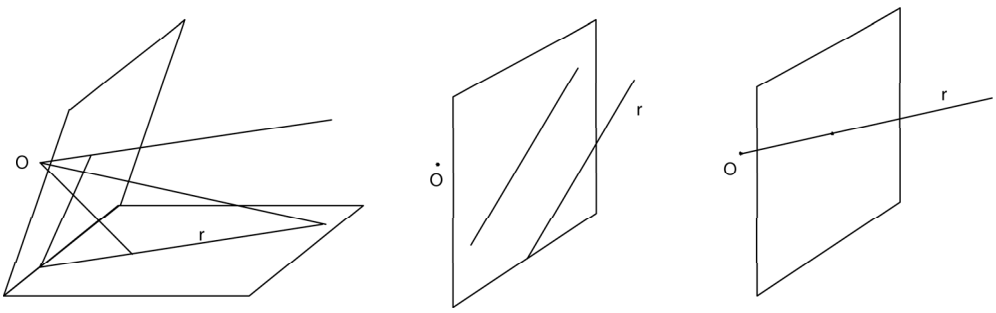


Figure 9.5: Projection of a 3D line: a) general case, b) parallel to image plane; c) containing the optical center

The perspective projection was discovered by the Greeks and later in Italy by the renaissance architects Brunelleschi and Leonardo da Vinci. It allows an accurate account of transformation produced by the eye when we observe a scene and wish to draw it.

9.6 Points, lines and planes

The projection of a point in space P onto the image plane is a point p whose coordinates are given by (9.20) (see Fig. 9.3). Let us now consider the projection of lines and planes. These projections were heavily used in painting and architecture since the Italian Renaissance with the works of Brunellesci and Leonardo da Vinci

The projection of a 3D line r on the image plane is a line or a point (see Fig. 9.5). The projection is a line if the line r does not contain the optical center (Fig. 9.5a,b). If the 3D line contains the optical center the projection is a point (Fig. 9.5c).

Let us discuss the first case and let us consider a point moving along the straight line. The projection of the point on the camera moves along the projected line and converges to a point called *vanishing point*. The vanishing point can is obtained by intersecting the image plane with a line parallel to r and containing the optical center O (see Figure 9.5a).

How can we compute the coordinates of the vanishing point? Let us consider a straight line r in the 3D space. The straight line is the set of all points

$$\mathbf{X} = \mathbf{X_0} + \alpha \mathbf{T} \tag{9.23}$$

where $\mathbf{X} = [X\ Y\ Z\ 1]^T$ is the position of a point on the straight line in homogeneous coordinates, $\mathbf{X_0} = [X_0\ Y_0\ Z_0\ 1]^T$ is the initial point, $\mathbf{T} = [T_x\ T_y\ T_z\ 0]^T$ is the tangent vector and α is a free

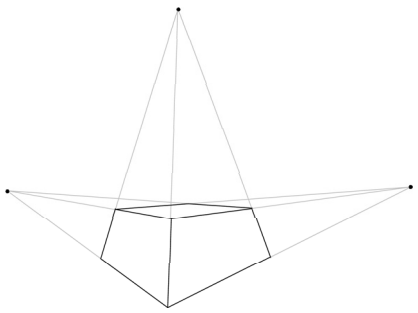


Figure 9.6: Projection of a cubic house using 3 vanish points.

parameter (time). Note that the fourth coordinate of $\mathbf{T}$ is zero. The projection of $\mathbf{X}$ on the image plane is (see (9.20))

$$\mathbf{x} = \mathbf{\Pi X}_0 + \alpha \mathbf{\Pi T} \tag{9.24}$$

The point $\mathbf{x}$ belongs to the straight line defined by the points $\mathbf{x}_1 = \mathbf{\Pi X}_0, \mathbf{x}_2 = \mathbf{\Pi T}$ (in homogeneous coordinates). When α tends to infinity

$$\mathbf{x} = \mathbf{\Pi T} \tag{9.25}$$

The vanishing point in Cartesian coordinates is given by

$$x = \frac{\pi_1^T T}{\pi_3^T T} \qquad y = \frac{\pi_2^T T}{\pi_3^T T} \tag{9.26}$$

with $\mathbf{T} = [T_x\ T_y\ T_z\ 0]^T$. Several conclusions can be draw. The vanishing point does not depend on the position of the line in space ($\mathbf{X_0}$). It only depends on the orientation. Therefore a set of parallel lines has the same vanishing point. This is a well known property of the perspective projection. There is one vanishing point for each set of parallel lines. For example, if we want to draw a building with cubic shape we must consider three vanishing points. Furthermore, the vanishing point of all lines belonging to a plane are located on an image line called the horizon (Fig. 9.6).

The projection of a plane is a plane except if the plane contains the projection center. In this case the projection is a straight line. Let us briefly discuss the horizon line we observe when we take a photograph of the sea. Is the horizon caused by the curvature of the earth (see Fig. 9.7)? Try to convince yourself that you would see a line of horizon even the case was plane.

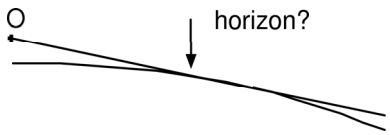


Figure 9.7: Is the line of horizon produced by the curvature of earth.

9.7 Camera Calibration

An important issue concerns the estimation of the camera parameters from an image. This process is known as the camera calibration.

Camera calibration is often done using a calibration object with with known geometry (see figure 9.8). The calibration object should contain a set of easily detected points. We then obtain an image of the calibration object and accurately detect all the calibration points. The calibration data is a set of 3D points and their projections

$$\mathcal{T} = \{(\mathbf{X}^i, \mathbf{x}^i), i = 1, \ldots, n\} \tag{9.27}$$

where $\mathbf{X}^i \in \mathbb{R}^3, \mathbf{x}^i \in \mathbb{R}^2$ are the coordinates of the i-th point in space and in the image plane.

The camera parameters can be chosen in order to minimize the distance from the points projected by the camera and by the model. This can be done minimizing the quadratic cost functional

$$E = \sum_{i=1}^{n} \|\mathbf{x}^i - f(\mathbf{X}^i, \pi)\|^2 \tag{9.28}$$

where $f : \mathbb{R}^3 \to \mathbb{R}^2$ is the perspective projection (9.29) and π are the model parameters. The minimization of (9.28) is a non convex optimization problem which can be numerically solved. However, the cost function has many local minima and the optimization algorithm can be trapped by one of them.

To overcome this difficulty simpler methods are used based on a different optimization criterion. The camera model (9.29) can be written as follows

$$x\mathbf{X}^T\pi_3 = \mathbf{X}^T\pi_1 \qquad y\mathbf{X}^T\pi_3 = \mathbf{X}^T\pi_2 \tag{9.29}$$

where π_i is the i-th line of matrix $\mathbf{\Pi}$.

Using matrix notation we obtain,

$$\begin{bmatrix} \mathbf{X}^T & \mathbf{0} & -x\mathbf{X}^T \\ \mathbf{0} & \mathbf{X}^T & -y\mathbf{X}^T \end{bmatrix} \pi = \mathbf{0} \tag{9.30}$$

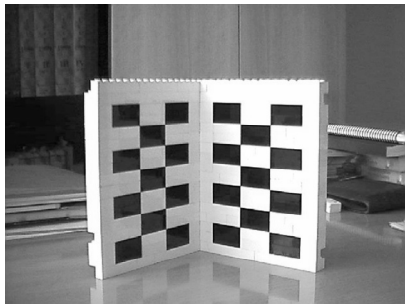


Figure 9.8: Calibration object

where $\pi \in \mathbb{R}^{9\times 1}$ is a vector with the coefficients of matrix $\mathbf{\Phi}$.

Each pair of points $(\mathbf{X}, \mathbf{x})$ leads to 2 linear equations. Therefore, the data set $\mathcal{T}$ defines a system of $2n$ equations

$$\mathbf{M}\pi = 0 \tag{9.31}$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{X}^{1T} & \mathbf{0} & -x^1\mathbf{X}^{1T} \\ \mathbf{0} & \mathbf{X}^{1T} & -y^1\mathbf{X}^{1T} \\ \vdots & & \vdots \\ \mathbf{X}^{nT} & \mathbf{0} & -x^n\mathbf{X}^{nT} \\ \mathbf{0} & \mathbf{X}^{nT} & -y^n\mathbf{X}^{nT} \end{bmatrix} \qquad \mathbf{M} \in \mathbb{R}^{2n\times 9} \tag{9.32}$$

We conclude from (9.31) that π should belong to the null space of $\mathbf{M}$. In practice $\mathbf{M}$ has rank 9 and π is obtained solving (9.31) by least squares. This is done minimizing the norm of the residue

$$\mathbf{e} = \mathbf{M}\pi \tag{9.33}$$

assuming that the camera coefficients are normalized i.e., $\|\pi\| = 1$. This is an optimization problem with constraints. We can solve this problem using a Lagrangean function

$$L = \pi^T\mathbf{M}^T\mathbf{M}\pi + \lambda(\mathbf{e}^T\mathbf{e} - 1) \tag{9.34}$$

The minimization of L can be done computing the derivative and making it equal to zero. We obtain

$$\mathbf{M}^T\mathbf{M}\pi = -\lambda\pi \tag{9.35}$$

Therefore π must be an eigen vector of $\mathbf{M}^T\mathbf{M}$. It can be easily shown that π is the eigenvector associated to the smallest eigen value of $\mathbf{M}^T\mathbf{M}$. The estimation of π is done by simply computing the eigendecomposition of matrix $\mathbf{M}^T\mathbf{M} \in \mathbb{R}^{9\times 9}$.

After obtaining $\mathbf{\Pi}$ we still need to obtain the intrinsic and extrinsic parameters of the camera. This is possible and can easily be done. Let $\mathbf{\Pi} = [\mathbf{A}\ \mathbf{b}]$ where $\mathbf{A}$ is a square matrix formed by the first three columns of $\mathbf{\Pi}$ and $\mathbf{b}$ is the las column. Therefore,

$$\mathbf{A} = \mathbf{KR} \qquad \mathbf{b} = \mathbf{KT} \tag{9.36}$$

where $\mathbf{K}$ is an upper triangular matrix and $\mathbf{R}$ is a rotation matrix.

The QR decomposition from Linear Algebra solves a similar problem. It decomposes a square matrix $\mathbf{M}$ into the product

$$\mathbf{M} = \mathbf{QR} \tag{9.37}$$

where $\mathbf{Q}$ is a rotation matrix and $\mathbf{R}$ an upper triangular matrix. We cannot use this result directly because the upper triangular and the rotation matrix appear in a different order. This difficulty is overcome by inverting $\mathbf{A}$. Therefore,

$$\mathbf{A}^{-1} = \mathbf{R}^{-1}\mathbf{K}^{-1} \tag{9.38}$$

where $\mathbf{R}^{-1}$ is a rotation matrix and $\mathbf{T}^{-1}$ is an upper triangular matrix[5]. We can apply the QR decomposition to $\mathbf{A}^{-1}$ to obtain $\mathbf{R}^{-1}, \mathbf{K}^{-1}$. The translation vector is then easily computed $\mathbf{T} = \mathbf{K}^{-1}\mathbf{b}$. The intrinsic and extrinsic (pose) parameters of the camera are easily obtained in this way.

[5]the inverse of a rotation matrix is still a rotation matrix and the inverse of an upper triangular matrix is still upper triangular

Bibliography

[1] Y. Ma, S. Soatto, J. Kosecka, S. Sastry, An Invitation to 3D Vision: From Images to Geometric Models, Springer Verlag, 2003.

[2] R. I. Hartley, A. W. Zisserman, Multiple View Geometry in Computer Vision, Second Edition, Cambridge University Press, 2001.

[3] I. Reid, A. Zisserman, Goal Directed Metrology, European Conference on Computer Vision, 1996.

Chapter 10

Shape and Motion Estimation

10.1 Introduction

Cameras project the information of 3D world into 2D images. There is loss of information in this transformation (depth and occlusions). Can we invert the projection performed by the camera and recover 3D shape and motion? This is the question addressed in this chapter and it is fundamental issue if we want to use cameras to measure the world.

The main question we will address is: *can we recover the 3D geometry and motion from images produced by a moving camera?*

It is difficult to answer this question using a single image. The scene points are projected onto the image plane by a perspective projection (Figure 9.3). The projection of a 3D point P defines an optical ray but does not convey any information about depth. Therefore, the projection is not enough to reconstruct P[1].

Suppose we have two views of the same scene with overlap (some points are visible in both views, see Figure 10.1). Knowing the projections p_1, p_2 of a point P in both views and knowing the optical centers of the cameras O_1, O_2, it is possible to recover P computing the intersection of the lines O_1p_1, O_2p_2 (see figure 10.2). This procedure is called **triangulation**.

We can repeat this procedure for many 3D points and estimate the objects surfaces by interpolation. This approach is rather general since it make no assumptions about the object shape. However it has some limitations. It requires accurate models of the cameras and assumes that we can detect many pair of corresponding points in both cameras. These restrictions can

[1] in some cases, depth can be recovered from a single image (e.g., if P belongs to a known plane)

Figure 10.1: Pair of images

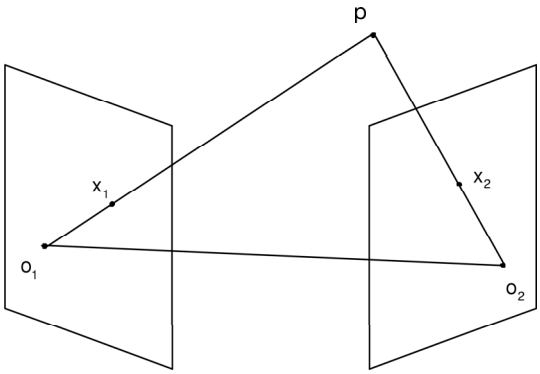


Figure 10.2: Triangulation: point P is obtained by intersecting the lines p_1O_1, p_2O_2

be aleviated using dense approach but we will not pursue that line here.

The estimation of a 3D point requires information about the intrinsic parameters of the camera and the camera motion. Three cases can be considered: i) known internal model and camera motion; ii) known internal model and unknown camera motion; iii) unknown internal model and camera motion. Table 10.1 summarises these three cases.

The first two problems are known as 3D reconstruction with calibrated camera. The last problem is more difficult and it is known as the uncalibrated reconstruction.

10.2 Epipolar Geometry

The reconstruction of P is obtained by intersecting the corresponding optical rays (see Figure 10.2). However, they may not intersect in space. This happens very often since the image

Available information	Information to be estimated
Intrinsic parameters motion	shape
intrinsic parameters	shape and motion
	shape and motion

Table 10.1: Hierarchy of reconstruction problems

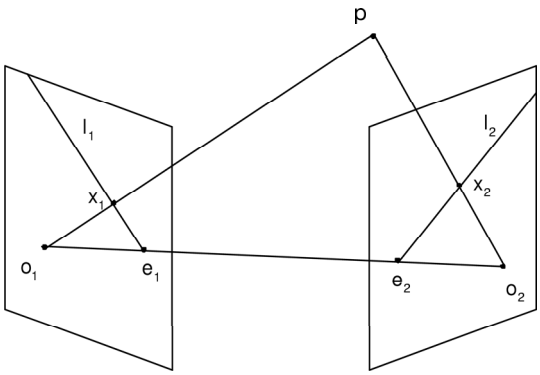


Figure 10.3: Epipolar geometry

coordinates are corrupted by noise and the camera model is approximate. This raises a question: *how can we guarantee that two optical rays associated to a pair of views intersect in space?*

The answer to this question is based on the concept of epipolar geometry and involves three concepts: epipolar plane, epipolar line and epipoles.

The optical centers O_1, O_2, p and the scene point P define a plane in 3D space known as the **epipolar plane** associated to P. The optical rays belong to the epipolar plane. Furthermore, the epipolar plane intersects the image planes in two lines l_1, l_2 known as **epipolar lines**. Therefore, each point P in space defines a pair of epipolar lines. These lines have an important role: two points p_1, p_2 are projections of a point P if and only if they belong to corresponding epipolar lines. Therefore, if we search for a pair of corresponding points we should restrict the search to pairs of epipolar lines. This allows important computational savings.

There are two points which belong to all the epipolar lines. These points are known as **epipoles**. The epipoles $\mathbf{e}_1, \mathbf{e}_2$ are obtained by intersecting the line O_1O_2 with the image planes. There are degenerated cases in which the epipolar plane is not uniquely defined. If $O_1 = O_2$ (pure rotation) there are an infinite number of epipolar planes.

We will now show that the epipolar geometry can be estimated from a pair of images without

knowledge about the camera motion or the 3D scene.

10.3 Essential Matrix

Let us for the sake of simplicity that the cameras are calibrated. This means that we know the intrinsic parameters of the camera. The projected points $\mathbf{x}_1, \mathbf{x}_2$ can be pre-processed in order to compensate the intrinsic parameters of the camera. Pre-processing is done by multiplying the homogeneous coordinates of the points by the inverse of matrix $\mathbf{K}$: $\mathbf{x} \rightarrow \mathbf{K}^{-1}\mathbf{x}$.

Under this hypothesis, the camera model in both positions is very simple

$$\lambda_1 \mathbf{x}_1 = \mathbf{X}_1 \qquad \lambda_2 \mathbf{x}_2 = \mathbf{X}_2 \tag{10.1}$$

where $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^3$ are the homogeneous coordinates and $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^3$ are the Cartesian coordinates of P measured in the cameras frames. Since $\mathbf{X}_2 = \mathbf{R}\mathbf{X}_1 + \mathbf{T}$ where $(\mathbf{R}, \mathbf{T})$ are the parameters of a rigid body transform, we obtain

$$\lambda_2 \mathbf{x}_2 = \mathbf{R}\lambda_1 \mathbf{x}_1 + \mathbf{T} \tag{10.2}$$

This equation creates a constraint on the projected points $\mathbf{x}_1$, $\mathbf{x}_2$. It is not easy to interpret this equation since it depends on the scale factors λ_1, λ_2 which are unknown. We will modify the equation to get rid of the scale factors.

Let us compute the external product of both members of (10.2) with $\mathbf{T} = [t_1 t_2 t_3]^T$. This can be done by multiplying both members by the skew symmetric matrix

$$\hat{\mathbf{T}} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{10.3}$$

We obtain

$$\lambda_2 \hat{\mathbf{T}}\mathbf{x}_2 = \lambda_1 \hat{\mathbf{T}}\mathbf{R}\mathbf{x}_1 \tag{10.4}$$

since $\hat{\mathbf{T}} \times \mathbf{T} = \mathbf{0}$. If we compute the inner product of both members of the equation by $\mathbf{x}_2$ we obtain

$$\hat{\mathbf{x}}_2^T \hat{\mathbf{T}}\mathbf{R}\mathbf{x}_1 = 0 \tag{10.5}$$

since $\hat{\mathbf{T}}\mathbf{x}_2$ is orthogonal to $\mathbf{x}_2$. This is a constraint that each pair of projected points $(\mathbf{x}_1, \mathbf{x}_2)$ must obey. This is also a sufficient condition for guarantying that two points in the image planes of both cameras are the projections of the same 3D point P.

The matrix $\mathbf{E} = \hat{\mathbf{T}}\mathbf{R}$ is called the essential matrix and plays an important role in 3D reconstruction from pairs of images. Furthermore, the matrix $\mathbf{E}$ can be directly estimated from a pair of images and allows us to compute epipolar geometry: epipolar lines and epipoles. from a pair of images and allows us to compute epipolar geometry: epipolar lines and epipoles.

Before we do this we need to see how to define a straight line in homogeneous coordinates. A straight line is defined by the equation

$$\mathbf{l}^T\mathbf{x} = 0 \tag{10.6}$$

where $\mathbf{l} \in \mathbb{R}^3$ is a vector containing the unit norm and the distance to the origin. The epipolar lines can now be easily computed from the essential matrix (10.5). If we keep $\mathbf{x}_1$ ($\mathbf{x}_2$) constant we obtain the equation of a line for the variable $\mathbf{x}_2$ ($\mathbf{x}_1$). This is the epipolar line. Both epipolar lines are defined by

$$\mathbf{l}_1^T\mathbf{x}_1 = 0 \qquad \mathbf{l}_2^T\mathbf{x}_2 = 0 \qquad \mathbf{l}_1 = \mathbf{E}^T\mathbf{x}_2 \qquad \mathbf{l}_2 = \mathbf{E}\mathbf{x}_1 \tag{10.7}$$

The epipoles can be easily computed as well. They must belong to all epipolar lines. Therefore,

$$\mathbf{E}\mathbf{e}_1 = 0 \qquad \mathbf{E}^T\mathbf{e}_2 = 0 \tag{10.8}$$

The epipole $\mathbf{e}_1$ is associated to the null space of the essential matrix $\mathbf{E}$ and the epipole $\mathbf{e}_2$ is associated to the null space of $\mathbf{E}^T$. Each of them can be easily computed from $\mathbf{E}$. If we use the equation $\mathbf{E} = \hat{\mathbf{T}}\mathbf{R}$ we can also conclude that $\mathbf{e}_1 = \mathbf{R}^T\mathbf{T}$ e $\mathbf{e}_2 = \mathbf{T}$ (apart from a scaling factor).

Matrix $\mathbf{E} = \hat{\mathbf{T}}\mathbf{R}$ has interesting properties. It can be shown that $\mathbf{E}$ is an essential matrix iif it has a null eigen value and the other two are equal [1]. The eigen decomposition of $\mathbf{E}$ is given by

$$\mathbf{E} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\mathbf{T} \qquad \mathbf{\Lambda} = diag\{\sigma, \sigma, 0\} \tag{10.9}$$

The space of matrices which verify this equation is called the *essential space*.

We will now consider an important question: is it possible to invert the application $(\mathbf{R}, \mathbf{T}) \rightarrow \mathbf{E} = \hat{\mathbf{T}}\mathbf{R}$. The answer is yes apart from the fact that there are two possible solutions for the inverse transform. Let $\mathbf{E}$ be a matrix belonging to the essential space, then $\mathbf{E} = \hat{\mathbf{T}}\mathbf{R}$ with:

$$\hat{\mathbf{R}}_1 = \mathbf{U}\mathbf{R}_\mathbf{z}(+\frac{\pi}{\mathbf{2}})\mathbf{\Sigma}\mathbf{U}^T \qquad \mathbf{T}_1 = \mathbf{U}\mathbf{R}_\mathbf{z}(+\frac{\pi}{\mathbf{2}})\mathbf{V}^T \tag{10.10}$$

$$\hat{\mathbf{R}}_2 = \mathbf{U}\mathbf{R}_\mathbf{z}(-\frac{\pi}{\mathbf{2}})\mathbf{\Sigma}\mathbf{U}^T \qquad \mathbf{T}_2 = \mathbf{U}\mathbf{R}_\mathbf{z}(-\frac{\pi}{\mathbf{2}})\mathbf{V}^T \tag{10.11}$$

where $\mathbf{R}_z(\theta)$ is a rotation of amplitude θ with rotation axis z

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10.12}$$

This result has an important impact in practical applications. **Motion information can be recovered from the essential matrix!**

10.4 Estimation of the Essential Matrix

How is the essential matrix estimated?

The essential matrix can be estimated from a pair of images by linear estimation algorithms.

The fundamental equation (10.5) is linear in the parameters of $\mathbf{E}$. Defining $\mathbf{x}_1 = [x_1, y_1, z_1], \mathbf{x}_2 = [x_2, y_2, z_2]$ we obtain

$$[x_1x_2 \;\; x_1y_2 \;\; x_1z_2 \;\; y_1x_2 \;\; y_1y_2 \;\; y_1z_2 \;\; z_1x_2 \;\; z_1y_2 \;\; z_1z_2\,]\,\mathbf{e} = 0 \tag{10.13}$$

where $\mathbf{e} = [e_{11}\; e_{21}\; \ldots e_{33}]^T$ is a vector with the elements of the essential matrix. The previous equation can be written as follows

$$\mathbf{a^T e = 0} \qquad \mathbf{a = x_1 \otimes x_2} \tag{10.14}$$

where $\otimes$ stands for Kronecker multiplication of two matrices [2].

Each pair of points defines a linear equation for $\mathbf{e}$. If we know n pairs of points detected in both images $\{(\mathbf{x_1^i}, \mathbf{x_2^i},)\}$ we obtain a system of equations

$$\mathbf{Me = 0} \tag{10.15}$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{x}_1^1 \otimes \mathbf{x}_2^1 \\ \vdots \\ \mathbf{x}_1^n \otimes \mathbf{x}_2^n \end{bmatrix} \quad , \qquad \mathbf{M} \in \mathbb{R}^{n \times 9} \tag{10.16}$$

Vector $\mathbf{e}$ belongs to the null space of $\mathbf{M}$.

Since $\mathbf{e}$ is defined up to a scale factor we assume that $||\mathbf{e}\$ = 1$. It is not possible to solve equation (10.16) in an exact way with the constraint. If we have more than 8 points, matrix

[2] $M = \mathbf{A} \otimes \mathbf{B}$ is a block matrix $\mathbf{M} = [\mathbf{M}_{ij}]$ where each block is the product of an element of $\mathbf{A}$ by $\mathbf{B}$ as follows $\mathbf{M}_{ij} = a_{ij}\mathbf{B}$.

$\mathbf{M}$ will have rank 9 and the null space has a single element $\mathbf{0}$).To overcome this difficulty we solve the problem using the least squares method by minimizing the sum of the residues

$$E = ||\mathbf{Me}||^2 \tag{10.17}$$

with the constraint $||\mathbf{e}||^2 = 1$. This problem can be solved using the Lagrangean function

$$E = ||\mathbf{Me}||^2 - \lambda(||\mathbf{e}||^2 - 1) \tag{10.18}$$

$$E = \mathbf{e^T M^T Me} - \lambda(\mathbf{e^T e} - 1) \tag{10.19}$$

Computing the derivative of E with respect to $\mathbf{e}$ we obtain

$$\mathbf{M^T Me} - \lambda\mathbf{e} = 0 \quad \Rightarrow \quad \mathbf{M^T Me} = \lambda\mathbf{e} \tag{10.20}$$

this equation is verified if e is an eigenvector of $\mathbf{M^T M}$. The minimum of E is achieved if we choose the eigenvector associated to the smallest eigenvalue of $\mathbf{M^T M}$.

In general, the least squares estimate is not a valid essential matrix since it does not belong to the essencial space. The next step consists of approximating the least squares estimate by a valid essential matrix. This is done projecting the estimate onto the essential space. Let the SVD decomposition of $\mathbf{E}$ be

$$\mathbf{E} = \mathbf{U\Lambda V}^T \qquad \mathbf{\Lambda} = diag\{\sigma 1, \sigma 2, \sigma 3\} \tag{10.21}$$

The projection of E on the essential space is

$$\mathbf{E}' = \mathbf{U\Lambda}'\mathbf{V}^T \qquad \mathbf{\Lambda}' = diag\{\sigma, \sigma, 0\} \tag{10.22}$$

em que $\sigma = (\sigma_1 + \sigma_2)/2$.

This algorithm is known as the 8 point algorithm since it requires at least 8 pairs of points. This algorithms was proposed by Longuet-Higgins (Psichologist, Cambridge) in the scope of the study of human perception and it is summarized in Table 10.2. This algorithm is very sensitive to measurement errors but can be significantly improved if the data is preprocessed. The mass center of the image points should be translated to the origin and the variance should be normalized.

10.5 Shape and Motion estimation (Calibrated camera)

Motion estimation can be done using the essential matrix $\mathbf{E}$ using (10.10). We will now focus on the estimation of shape of the objects viewed by both cameras. We will assume that all the

8 point algorithm (Longuet-Higgins)

1. **Data acquisition:** determine N pairs of corresponding points in both cameras: $\{(\mathbf{x}_1^i, \mathbf{x}_2^i), i = 1, \ldots, N\}$ $(N \geq 8)$

2. **Pre-processing:** center and normalize each set of points $\mathbf{x}_1^i, \mathbf{x}_2^i$ subtracting their mean and scaling them by the inverse standard deviation. These operations will be denoted by $\mathbf{A}_1, \mathbf{A}_2$.

3. **Least squares estimate:** compute the eigenvector associated to the smallest eigenvalue of $\mathbf{M}^T\mathbf{M}$. Matrix $\mathbf{M}$ is defined in (10.16). This estimated is denoted by $\mathbf{E}$

4. **Projection:** determine the SVD of E $\mathbf{E} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$,
$\mathbf{\Lambda} = diag\{\sigma 1, \sigma 2, \sigma 3\}$ and compute the projection of $\mathbf{E}$ the essencial subspace
$\mathbf{E}' = \mathbf{U}\mathbf{\Lambda}'\mathbf{V}^T$, em que $\mathbf{\Lambda}' = diag\{\sigma, \sigma, 0\}$ e $\sigma = (\sigma_1 + \sigma_2)/2$.

5. **Post-processing:** $\hat{\mathbf{E}} = \mathbf{A}_2^T\mathbf{E}'\mathbf{A}_1$

Table 10.2: 8 point algorithm for the estimation of E

3D points to be reconstructed are independent and will be estimated in an independent way[3].

There are several ways to estimate P from the projections $\mathbf{x}_1, \mathbf{x}_2$. They are all equivalent when the cameral model is perfect and the observation noise is zero. However, the methods perform in different ways when applied to real data. We will now discuss a simple linear algorithm to estimate the 3D position of observed points.

If we adopt the model (10.1) we conclude that the estimation of P is equivalent to the estimation of λ_1 or λ_2. In general it i not possible to enforce both conditions in an exact way. We prefer instead to define algebraic errors $\epsilon_1, \epsilon_2 \in \mathbb{R}^3$ between both members of the equations

$$\epsilon_1 = \lambda_1 \mathbf{x}_1 - \mathbf{X} \qquad \epsilon_2 = \lambda_2 \mathbf{x}_2 - (\mathbf{R}\mathbf{X} + \mathbf{T}) \tag{10.23}$$

Since it is not possible to guarantee that all the errors are zero, the vector $\mathbf{X}$ is estimated by minimizing a quadratic cost functional

$$E = \|\lambda_1 \mathbf{x}_1 - \mathbf{X}\|^2 + \|\lambda_2 \mathbf{x}_2 - \mathbf{R}\mathbf{X} - \mathbf{T}\|^2 \tag{10.24}$$

where $\|.\|$ is the Euclidean norm. We will minimize

$$E = (\lambda_1 \mathbf{x}_1 - \mathbf{X})^T (\lambda_1 \mathbf{x}_1 - \mathbf{X}) + (\lambda_2 \mathbf{x}_2 - \mathbf{R}\mathbf{X} - \mathbf{T})^T (\lambda_2 \mathbf{x}_2 - \mathbf{R}\mathbf{X} - \mathbf{T}) \tag{10.25}$$

A necessary condition verified by the minima of E is

$$\frac{dE}{d\mathbf{X}} = 0 \tag{10.26}$$

Computing the derivative of E with respect to $\mathbf{X}$ (see Appendix)

$$-(\lambda_1 \mathbf{x}_1 - \mathbf{X}) - \mathbf{R}^T (\lambda_2 \mathbf{x}_2 - \mathbf{R}\mathbf{X} - \mathbf{T}) = 0 \tag{10.27}$$

$$(\mathbf{I} + \mathbf{R}^T \mathbf{R})\mathbf{X} = \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{R}^T \mathbf{x}_2 - \mathbf{R}^T \mathbf{T} \tag{10.28}$$

This is a system of 3 linear equations. The solution is given by

$$\mathbf{X} = (\mathbf{I} + \mathbf{R}^T \mathbf{R})^{-1} (\lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{R}^T \mathbf{x}_2 - \mathbf{R}^T \mathbf{T}) \tag{10.29}$$

The matrix inversion can be done once for all the data points since it does not depend on the points detected in the images.

This method is algebraic and the cost functional does not have a geometric meaning. To overcome this difficulty we could reconstruct P by estimating the point closest to the straight lines $o_1 x_1$, $o_2 x_2$. This method is also linear.

[3]some methods take the opposite approach and try to estimate a dense depth map

Let us consider a third approach based on the image plane. Let us assume that the projected pints are corrupted by noise. Therefore the available measurements are

$$\mathbf{x}_1' = \mathbf{x}_1 + \mathbf{w}_1 \qquad \mathbf{x}_2' = \mathbf{x}_2 + \mathbf{w}_2 \tag{10.30}$$

where $\mathbf{w}_1, \mathbf{w}_2$ is the observation noise and $\mathbf{x}_1, \mathbf{x}_2$ are the true projections which should belong to a pair of epipolar lines and should verify (10.5). The reconstruction problem can be formulated as follows O problema: what is the pair of epipolar lines closest to $\mathbf{x}_1', \mathbf{x}_2'$? the answer to this question involves the solution of a nonlinear optimization problem.

Exercises

1. [Reconstrução 3D] Let x_1, x_2 be the projections of a point P by a calibrated camera at two positions and let $(\mathbf{R}, \mathbf{T})$ be the motion parameters. Compute the coordinated of the space point $\mathbf{X}$ closest to the straight lines o_1x_1, o_2x_2. (Hint: use (10.1) and motion transformation)

Bibliography

[1] Y. Ma, S. Soatto, J. Kosecka, S. Sastry, An Invitation to 3D Vision: From Images to Geometric Models, Springer Verlag, 2003.

[2] R. I. Hartley, A. W. Zisserman, Multiple View Geometry in Computer Vision, Second Edition, Cambridge University Press, 2001.

[3] J. P. Costeira, T. Kanade, A Multibody Factorization Method for Independent Moving Objects, International Journal on Computer Vision, 29(3), pp.159-179,

Apêndices

Derivadas em ordem a matrizes

A derivada de uma função escalar $f(\mathbf{X})$ em ordem a uma matriz $\mathbf{X}$ de dimensões $n \times m$ é uma matriz definida por

$$\left(\frac{\mathbf{df}}{\mathbf{dX}}\right)_{ij} = \frac{\partial f}{\partial X_{ij}} \tag{10.31}$$

Nalguns textos a matriz de derivadas aparece transposta em relação a esta definição aqui adoptada.

Em seguida mencionam-se algumas propriedades do traço e do determinante de uma matriz admitindo-se que $\mathbf{X}$, $\mathbf{A}$ e $\mathbf{B}$ são matrizes de dimensões apropriadas.

Propriedades do traço:

$$\frac{d}{d\mathbf{X}} tr\{\mathbf{X}\} = \mathbf{I}^T \tag{10.32}$$

$$\frac{d}{d\mathbf{X}} tr\{\mathbf{AXB}\} = \mathbf{A}^T\mathbf{B}^T \tag{10.33}$$

$$\frac{d}{d\mathbf{X}} tr\{\mathbf{AX}^T\mathbf{B}\} = \mathbf{BA} \tag{10.34}$$

$$\frac{d}{d\mathbf{X}} tr\{\mathbf{XX}^T\} = 2\mathbf{X} \tag{10.35}$$

$$\frac{d}{d\mathbf{X}} tr\{\mathbf{AXBX}\} = \mathbf{A}^T\mathbf{X}^T\mathbf{B}^T + \mathbf{B}^T\mathbf{X}^T\mathbf{A}^T \tag{10.36}$$

$$\frac{d}{d\mathbf{X}} tr\{\mathbf{AXBX}^T\} = \mathbf{AXB} + \mathbf{A}^T\mathbf{XB}^T \tag{10.37}$$

$$\frac{d}{d\mathbf{X}} tr\{\mathbf{AX}^{-1}\mathbf{B}\} = -\mathbf{X}^{-1}\mathbf{BAX}^{-1} \tag{10.38}$$

$$\frac{d}{d\mathbf{X}} |\mathbf{X}| = |\mathbf{X}|\, \mathbf{X}^{-\mathbf{T}} \tag{10.39}$$

$$\frac{d}{d\mathbf{X}} \log|\mathbf{X}| = \mathbf{X}^{-\mathbf{T}} \tag{10.40}$$

$$\frac{d}{d\mathbf{X}} |\mathbf{AXB}| = |\mathbf{AXB}|\, \mathbf{X}^{-\mathbf{T}} \tag{10.41}$$

$$\frac{d}{d\mathbf{X}} |\mathbf{X}|^n = n|\mathbf{X}|^n\, \mathbf{X}^{-\mathbf{T}} \tag{10.42}$$